University of Zurich[UZH]

Department of Informatics

A Real-Time Vision-Based Mobile Robot System

A dissertation submitted to the Faculty of Economics, Business Administration and Information Technology of the University of Zurich

for the degree of
Doctor of Science (Ph.D.)

by
Thomas Hübner
from Germany

Accepted on the recommendation of

Prof. Dr. R. Pajarola
Prof. A. Bernstein, Ph.D.

2010

The Faculty of Economics, Business Administration and Information Technology of the University of Zurich herewith permits the publication of the aforementioned dissertation without expressing any opinion on the views contained therein.

Zurich, Oktober 27, 2010

The head of the Ph.D. program in informatics: Prof. Abraham Bernstein, Ph.D.

KURZFASSUNG

In dieser Dissertation wird eine Methode für die bildbasierte Echtzeitabbildung von Innenräumen mit einem mobilen Robotersystem vorgeschlagen. Dieses System verwendet kommerzielle Komponenten, die eine flexible Konstruktion bei geringen Kosten ermöglichen. Das entwickelte mobile Robotersystem, verfügt neben dem geringen Gewicht, einem differenzialen Fahrwerk und ausreichender Robustheit, über eine beträchtliche unabhängige Laufzeit, was eine notwendige Voraussetzung für die halbautomatische Navigation und Umgebungsabbildung ist. Bilddaten werden kontinuierlich, unterstützt von verschiedenen Sensoren und aktiver Beleuchtung, mit hochauflösenden Kameras erfasst. Die gesamte Datenverarbeitung wird direkt auf der mobilen Roboterplattform in Echtzeit durchgeführt.

Ein neues Software Framework wird eingeführt, das die Roboterhardware optimal nutzt und den herausfordernden Echtzeitbedingungen eines bildbasierten Systems genügt. Das neuentwickelte Framework enthält eine auf parallele Bild- und Datenverarbeitung optimierte Pipeline, die eine hochperformante Bildanalyse ermöglicht.

Der erste Teil der Dissertation legt den Fokus auf die Erfassung und Integration von Bildmerkmalen in Innenräumen. Die effiziente Erkennung und robuste Beschreibung von Bildmerkmalen in Innenräumen ist Aufgrund von fehlenden Strukturmerkmalen ein schwieriges Problem. Eine Lösung wird präsentiert, die eine schnelle angepasste Merkmalserkennung mit einer robusten skalierungsunabhängigen Merkmalsbeschreibung verbindet. Mit diesem System werden Bildmerkmale unabhängig von der Umgebung und Kameraauflösung mit einer kon-

stanten Geschwindigkeit erkannt und erfasst. Folglich ist die Zeit für die Merkmalsbeschreibung konstant und Echtzeitverarbeitung garantiert. Dieses System übertrifft deutlich die aktuellen Methoden für die Merkmalsdetektion und Merkmalsbeschreibung. Die Verwendung von optimierter räumlicher Merkmalszuordnung und nichtlinearer Optimierung, reduziert Ausreisser signifikant.

Dies erlaubt die präzise Abschätzung der Transformationen zwischen verschiedenen Kameraansichten, sowie der 3D-Position der Bildmerkmale. Basierend auf diesem Ansatz, werden Konzepte für die halbautomatische Navigation und Umgebungsabbildung abgeleitet.

Unzulänglichkeiten in der Mechanik des differenzialen Fahrsystems und verschiedene umgebungsabhängige Faktoren beeinflussen die Navigation. Ein schneller und einfacher Algorithmus für die Ausbalancierung des Fahrsystems gemäss dieser Faktoren wird zur Verfügung gestellt. Eine vorherige Kalibrierung ist nicht erforderlich. Im Gegensatz zu sensor-basierten Ansätzen, liefert der bildbasierte Algorithmus eine direkte Rückmeldung. Erkannte Bildmerkmale bestimmen den Expansionspunkt, der die Navigationsrichtung für eine Translationsbewegung angibt. Die Navigationsrichtung wird zur Erkennung und Korrektur der seitlichen Abweichung benutzt. Eine Schlüsselkomponente für die Navigation ist die Fähigkeit zur Ortsbestimmung. Die Position und Ausrichtung der mobilen Roboterplattform wird mit Hilfe von 2D Merkmalsübereinstimmungen innerhalb der Umgebung ermittelt. Die Präzision ist dadurch unabhängig von den ungenauen Tiefenwerten die mit 3D Bildmerkmalen verbunden sind. Translationsbewegungen werden zuverlässig von Bildmerkmalen senkrecht zur Fahrbewegung des Roboters bestimmt. Eine angepasste Merkmalsintergration bestimmt die Rotation der mobilen Roboterplattform. Die Fähigkeit zur Ortsbestimmung ist grundlegend für die Abbildung von Merkmalen und Hindernissen.

Die Hinderniserkennung und Hindernissumgehung kombiniert die Vorteile von zwei Systemen. Hindernisse werden in Echtzeit mit einem Distanzsensor erkannt, während ihre genaue Position und Form durch aktive Beleuchtung und Bildanalyse ermittelt wird. Dies unterstützt ebenfalls die Auswahl der richtigen Strategie zur Umgehung von Hindernissen.

Die Bilddaten der halbautonomen mobilen Roboterplattform werden benutzt, um eine grundlegende Umgebungskarte in Echtzeit zu erstellen. Dafür werden Bildmerkmale von verschiedenen Positionen innerhalb der Umgebung erfasst und integriert. Die so erstellte Umgebungskarte ermöglicht eine präzise Navigation und kann durch aktive Beleuchtung mit der 3D Rekonstruktion von Objekten im Nahbereich noch verfeinert werden.

ABSTRACT

In this dissertation, we propose a real-time vision-based mobile robot approach for the mapping of indoor environments. The system incorporates commercial off-the-shelf (COTS) components that allow a flexible system construction at low-cost. Besides its light weight, a differential drive system and the robustness, our mobile robot system provides an extensive independent runtime essential for semi-autonomous navigation and environment mapping. Visual data is continuously acquired with high-resolution cameras, supported by various sensors and active-illumination. The entire data processing is performed directly on the mobile robot platform in real-time.

To best exploit the robot hardware and satisfy the challenging real-time constraints of a vision-based system, we introduce a new software framework. This framework includes a multi-threaded image- and data-processing pipeline that enables high-performance image sensor analysis.

The first part of this dissertation focuses on the acquisition and integration of visual features in natural indoor environments. Efficient detection and robust description of visual features have proven to be extremely challenging in indoor environments because of a lack of dense texture. We present a solution to this problem that combines a fast adaptive feature detection stage with a stable scale-invariant feature description method. With our system, visual features are detected independently of the environment and the acquired resolution at a constant rate. Consequently, the description time is constant and real-time performance is guaranteed. Our proposed system clearly outperforms current standard methods for

feature detection and description. Utilizing optimized spatial matching and non-linear optimization, outliers are significantly reduced. This allows precise estimation of view transformations together with the 3D positions of the visual features. Based on this approach, we derive concepts for semi-autonomous navigation and environment mapping.

Imperfections in the mechanics of the differential drive system and environment dependent factors influence the navigation. We provide a fast and simple algorithm for balancing the drive system on-the-fly according these factors. Pre-calibration is not required. Unlike sensor-based approaches, our vision-based algorithm provides a direct feedback. Detected visual features determine the focus of expansion (FOE) that corresponds to the heading direction for a translational motion. This heading direction is used for locating and correcting lateral drift.

A key component for navigation is self-localization. To locate the position and orientation of the mobile robot platform inside the environment as precisely as possible, 2D feature correspondences are employed. Thus the precision is independent of the depth uncertainties associated with the visual features. Translational movements are reliably determined from visual features perpendicular to the driving direction of the robot. Adapted feature integration estimates the rotation of the mobile robot platform. Self-localization is essential for the mapping of features and obstacles.

Our obstacle detection and avoidance is combining the advantages of two systems. Obstacles are detected in real-time with a range-sensor, while their precise position and shape is obtained through vision under active-illumination. This further supports the selection of the right obstacle avoidance strategy.

Finally, visual data gathered by our semi-autonomous mobile robot platform is used to build a basic environment map in real-time. Visual features are acquired and integrated from different positions throughout the environment. The environment map enables accurate navigation and can be refined through active-illumination for the 3D reconstruction of close-range objects.

ACKNOWLEDGEMENTS

The Visualization and MultiMedia Lab at the University of Zurich provided a unique environment for my PhD work.

First of all, this is mainly thanks to my doctoral advisor Renato Pajarola. During the years, he gave me always his support and I could greatly benefit from his broad experience. The provided freedom was simply extraordinary, thus I will always look back on this time with great delight.

Secondly, many thanks go to my colleagues. I appreciated their contributions to my projects, but most thankful I'm for the fact that I could always motivate them for holidays, sport events and other activities. I hope that we will be able to stay in touch.

Finally, I am very grateful to my parents who always believed in me, whereby I could manage even the difficult times.

CONTENTS

NOTATIONS

Acronyms

AC	Alternating current
API	Application Program Interface
API	Application Programming Interface
ATX	Advanced Technology Extended
CCD	Charge-coupled device
COTS	Commercial off-the-shelf
DC	Direct current
DMA	Direct memory access
DPSS	Diode-pumped solid-state
EDR	Enhanced Data Rate
FIFO	First In First Out
FOE	Focus of expansion
GND	Ground
GPIO	General Purpose I/O
ICP	Iterative Closest Point
KLT	Kanade-Lucas-Tomasi Feature Tracker
KTP	Potassium titanyl phosphate ($KTiOPO_4$)
LCD	Liquid crystal display
LED	Light-emitting diode
Li-Ion	Lithium Ion

LSF	Least-square fitting
MOSFET	Metal-oxide-semiconductor field-effect transistor
MPV	Mid-point voltage
NiMh	Nickel Metal hydride
PCB	Protection Circuit Module
PSU	Power Supply Unit
PWM	Pulse-Width Modulation
R/C	Radio controlled
RANSAC	Random Sample Consensus
RMS	Root Mean Square
SBA	Sparse bundle adjustment
SIFT	Scale-invariant feature transform
SIMD	Single Instruction Multiple Data
SSD	Sum of Squared Differences
SSE	Streaming SIMD Extensions
SURF	Speeded Up Robust Features
SVD	Singular value decomposition

LIST OF FIGURES

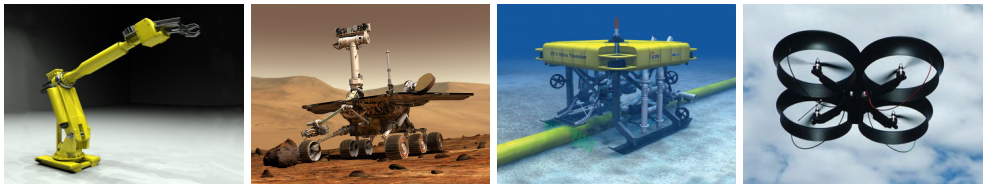
CHAPTER 1

INTRODUCTION

In recent years, there has been a tremendous increase of interest in robotics research. Nowadays, a similar trend is happening to robotics as we have seen for computers, the transition from large, heavy and expensive systems to small, light and low-cost embedded devices. This trend is driven by human necessities and the evolution of application fields. Industrial robots are still the most prevalent type of robots. They are extremely durable, designed to manipulate objects and capable of doing the same repetitive task over and over again. Present robotic systems have barely something in common with their industrial predecessors, as they have become mobile. Besides industry, mobile robot systems can be found in each and every domain ranging from wheeled service robots to submarines for underwater exploration and aerial vehicles for surveillance (Figure 1.1). Although shape, size and application area of a mobile robot system may differ, their principle characteristics are the same. Each mobile robot system is defined through these key components:

- Sensors and actuators

  Mobile robots interact with the environment through sensing and perception. Sensors are crucial to the operation in unknown and dynamic environments where it is impossible to have complete a priori information. Active components, the actuators, put the robot in motion. Simple actuators consist of electric motors and gears, cable drives, or chain drives. More sophisticated actuators involve the use of hydraulics, pneumatics, or magnetic interaction.



(a) (b) (c) (d)

Figure 1.1: *(a) Industrial robot. Image courtesy of TurboSquid. (b) Mars rover, Wheeled robot, Image courtesy of NASA. (c) Trencher, Autonomous Underwater Vehicle (AUV). Image courtesy of Soil Machine Dynamics. (d) CyberQuad, Unmanned Aerial Vehicle (UAV). Image courtesy of Cyber Technology.*

- Power electronics

  A mobile robot requires its own power source for autonomous operation. The power electronics provides the power and distributes it to each component of the robot system. Optionally, the power source can be monitored to determine the remaining operating time and current power consumption.

- Feedback control

  Feedback control is a technique by which a closed-loop system regulates itself. Sensory information is used directly to control the actuators of the system. In a system that uses feedback control to stabilize itself, there must be a tolerance between opposing functions to prevent dead-locks.

- Hardware/computer interface

  In a robotic system, the hardware interface is a device that connects the electronic parts with the controller, in the same way peripherals are connected to a computer system.

- Controller

  The controller is the ”brain” of the robotic system providing the necessary intelligence to control the robot, to process sensory information and to compute control commands for the actuators.

Mobility provides the robot with an enhanced operating capacity, opening a new area of research: The perception of the environment. Many current mobile robot systems make restrictive assumptions about the environment. In structured environments, the perception process allows the generation of maps or models of the world for localization and mapping. The main research in mobile robotics is focusing on unstructured and dynamic environments. In these environments, the

robot has to develop its own strategy for exploration using an adapted behavior and perception.

In the past mainly direct range sensing was used as a system to explore the environment. In nature, similar systems are employed only by a few types of nocturnal animals such as bats. Humans base many routinely performed tasks on visually perceived information. In order that a robotic system can perform tasks without extensive instrumentation or re-engineering of the environment, it must have the ability to perceive and behave depending on visual information. Therefore, vision is an important sensor for robotic systems imitating the human vision and allowing non-contact measurements of the environment.

To date, the heavy computational requirements involved in real-time image processing have hindered the widespread use of vision in mobile robotics, leaving the autonomous vision-based exploration of unstructured environments an unsolved research task.

1.1 Challenges of vision-based systems

The aim of the vision-based approach is to provide the mobile robot with certain abilities: The understanding of the environment, behavior adaptation, feedback control, navigation, self-localization and map building. Each of these abilities is associated with uncertainties and introduces challenging problems.

In the list below, we outline the major challenges of vision-based systems in the context of unstructured indoor environments.

1. Overall system cost

   Nowadays, the overall cost of a system is an important factor. For each hardware component, we have to find a compromise between the components price and its characteristics. In the case of a vision-based system, this requirement limits the resolution of the cameras and the quality of their optical system, leading to imperfections in the perception. Specialized vision processing hardware, while showing high-performance, is very expensive. Their universal counterparts are cheaper but require more sophisticated software.

2. Environment perception

   The lack of prior knowledge requires the visual perception of the environment. We need to determine the navigable space, recognize obstacles, estimate the position and shape of objects, identify dynamic parts under a constantly changing environment.

Vision-based approaches generally use visual features for this purpose. A visual feature is a clearly defined interest point with rich local information in image space. Visual features should expose a high degree of reproducibility and be stable under transformations (rotation, translation and scale) as well as brightness and illumination changes.

In an unstructured environment, visual sensing can be affected by the background, colors, textures, lighting variation, reflections, occlusions and a variety of other distractions. Therefore, we can not assume the accurate detection of visual features. The robust feature association is a significant challenge. Visual features need to be uniquely matched from various view-points in the environment. The problem is particularly difficult when the displacement of the camera between view-points is substantial. Tracking visual features is a problem similar to matching. However, in this case the displacement between the view-points is smaller. Visual feature tracking is more time consuming than the visual feature detection, as it involves a broad, unspecific correspondence search.

Visual feature integration is the final part of the environment perception. Associated visual features are unified and the mobile robot system can estimate the transformations between view-points. Errors in the previous association can lead to failures in the robots internal models. This problem is especially acute in unstructured environments, requiring a robust approach.

3. Real-time operation

   To interact and work inside an unstructured or even dynamic environment, the robot is required to perform the majority of tasks in real-time. With vision, most tasks are not performed in real-time due too the huge amount of image data that needs to be processed with the limited available resources on the mobile robot platform. They are limited by the physical dimensions of the robot platform and the electric power. The problem is usually solved using specialized signal processing hardware or off-platform parallel computers and networks.

4. Environment-adapted behavior and vision-based feedback control

   The adapted behavior of robots in an unknown environment is still an open issue. Mobile robots are typically tested in man-made or specifically prepared environments, whereby external impact parameters are known and the robot behavior is predefined. Alteration of the environment strongly affects the robot and can cause a fuzzy behavior. Hence environment-adapted behavior based on a regulatory vision-based feedback system is desirable.

5. Localization

   The localization process allows a mobile robot to know where it is at any moment in time relative to its environment. Localization can be local or global. Local localization, determines the robot position relative to an initial location, incrementally correcting the position, whereas in global localization the initial position is not needed. With visual perception, the localization process can be based on landmarks whose location is well known or automatically estimated. In unstructured and dynamic environments, the position of landmarks is usually unknown, introducing uncertainty in the localization.

6. Mapping

   In this context, mapping refers to the problem of creating a coherent environment representation from individual view-points. During mapping, the mobile robot has to detect and integrate landmarks, self-localize and integrate different visual information's. The mapping process is thus influenced by all previous steps from the environment perception up to the localization. Errors and uncertainties accumulate. Large amounts of data have to be processed and integrated to achieve a reasonable environment representation that can be employed for navigation and visualization. How information in the map is presented should depend on its significance and type.

1.2 Contributions

In this thesis, we solve some major problems of vision-based mobile robot systems. In particular, we address the areas 1-6 previously described in section 1.1. Figure 1.2 gives an overview over the contributions of this thesis.

Initially, we focus on the mobile robot platform providing the basis for our vision-based research. We pay specific attention to the several hardware components, bearing in mind stability, efficiency, and low cost. Second, we focus on data acquisition. Exploiting the limited available resources on our platform, we need to process acquired images and data in real-time and to detect and integrate natural features from the environment. Finally, we employ these features to provide our mobile robot platform with an environment-adapted behavior and localization and mapping functionality. Our contributions are as follows:

1. *Low-cost mobile robot system*

   We introduce a semi-autonomous mobile robot system that represents a reliable and robust platform for the development and testing of novel vision-based algorithms. The system is build from standard COTS components at

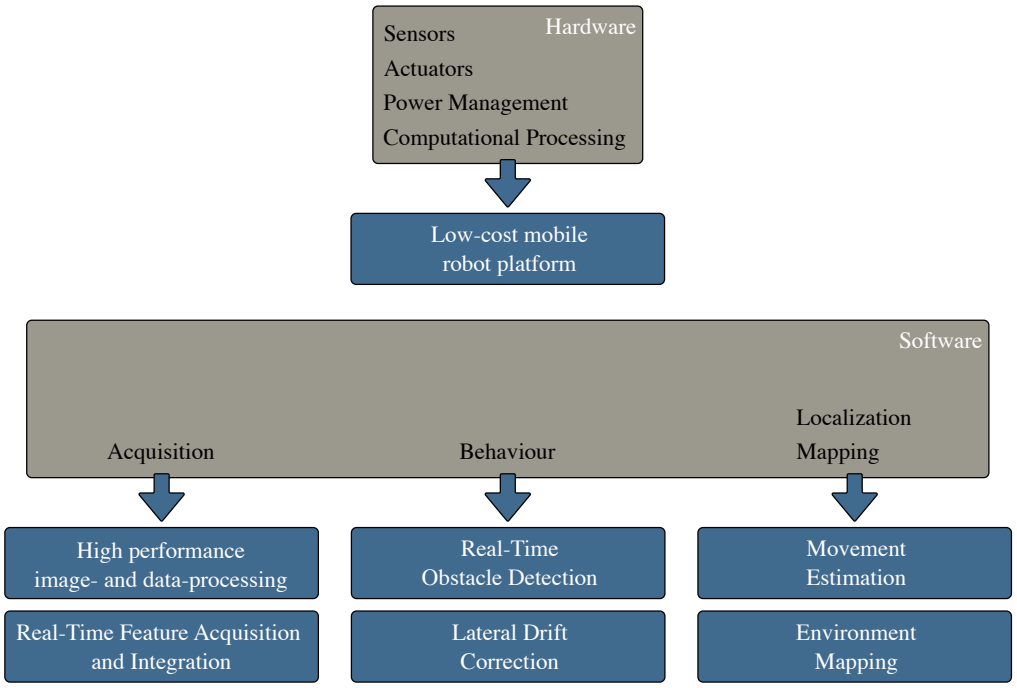


Figure 1.2: *Thesis focus.*

low overall cost. By focusing on low weight and low power consumption, we achieve a small, flexible system that exposes an exceptional autonomous operating time. Modularity ensures that components can be upgraded. Visual data from independent high-resolution cameras is processed in real-time on the robot system. Our computational processing unit employs a common operating system. Development, testing and debugging are therefore substantially simplified.

2. *High-performance image- and data-processing*

   To ensure real-time processing, we exploit current multi-core CPU architecture and SSE extensions. Our novel approach can efficiently acquire and process data. The visual processing pipeline ensures that available computational resources are optimally used and a minimal amount of processing time is spent on recurring image processing and analysis tasks. Furthermore, our mobile robot platform API abstracts actuators and sensors to provide a high-level programming interface.

3. *Feature acquisition and integration*

We present an efficient system for real-time feature acquisition and integration on COTS vision-based mobile robot platforms with limited computational resources. To overcome the performance limitations of current standard methods, we combine an adaptive feature detection based on the *FAST* corner detector with an improved *SIFT* based feature description. Our fast feature matching exploits spatial and temporal coherence. Features are stably and accurately integrated from a non-stationary stereo camera. Experimental comparisons show the advantages of our feature acquisition and integration and demonstrate its accuracy and robustness in natural indoor environments.

4. *Obstacle detection and lateral drift correction*

   A novel obstacle detection and avoidance system is proposed. The basic idea is to combine sonar-based long-range obstacle detection with a close-range vision-based refinement under active-illumination. This gives the advantage of a real-time detection together with a precise definition of the obstacles position and shape. Results show, that our obstacle detection is accurate and reliable. Possible problems are discussed in detail.

   To avoid lateral drift, we balance the robot's drive system with a vision-based feedback system. The lateral drift is determined and corrected in real-time based on the focus of expansion (FOE). Our methods accuracy is only limited by the camera resolution and the environment related feature quality.

5. *Localization and environment mapping*

   Our vision-based localization approach is adapted to indoor environments with sparse and unstable features. Translational and rotational movements are decoupled. Using the two-dimensional position of image features, acquired perpendicular to the robots movement direction, we obtain reliable, strong pronounced feature vectors and eliminate possible depth uncertainties. The results exhibit accuracy similar to commercial indoor localization systems.

   Using localization, we can integrate 3D points obtained through feature detection, stereo-vision or active-illumination into a single environment representation. We compare the resulting environment maps from the different methods and show that combining feature based environment mapping with active-illumination can generate a map sufficient for navigation and visualization.

1.3 Dissertation Overview

This dissertation first delivers a detailed insight into our mobile robot system in chapter 2. We discuss each individual component, from the basic robot platform, the drive system, the power management and the hardware interface up to the sensors, actuators and the computational processing unit. We employ only commercial off-the-shelf (COTS) components. In addition to the cost, components where chosen depending on their power consumption, weight and other characteristics. We describe how the mobile robot system is build from these components together with current issues and possible improvements. Electrical connections are illustrated as well as the required component modifications. Particular attention is given to the battery configuration to ensure an extensive system runtime and to secure it against faults.

Chapter 3 introduces a new software framework. This framework is adapted to the systems hardware components, enabling high-level programming access. We focus on image acquisition using different libraries and show how a multi-threaded pipeline approach can be utilized to process visual data efficiently in real-time on a multi-core processor architecture.

In chapter 4 we address the problem of feature acquisition and integration in unstructured indoor environments. Our solution combines a fast adaptive feature detection stage with a stable scale invariant feature description to detect and describe features in real-time on high-resolution images. The performance results are presented and compared to established methods. Our new method demonstrates superior performance and in addition achieves an excellent accuracy for the feature integration.

Chapter 5 describes some fundamental problems of navigation in an unstructured indoor environment. First of all, we develop a system to precisely determine the shape and position of obstacles in real-time. Artifacts preventing or interfering with our obstacle detection are discussed in detail. Secondly, we explain the estimation and correction of lateral drift that occurs during navigation. Finally, we propose a procedure for self-localization to estimate translational and rotational movements of the robot platform reliably.

Chapter 6 integrates all methods presented in this thesis to build an environment map that can be used for navigation and visualization. Results are shown for environment mapping with 3D features, 3D stereo and active-illumination. The fusion of 3D feature mapping and active-illumination delivers promising results, defining the direction for future research.

Each chapter 4-6 includes a section about relevant related work and experimental results.

We conclude the dissertation in Chapter 7 with a summary and directions for future work.

CHAPTER 2

ROBOTICS HARDWARE

Our semi-autonomous robot is based on the MMP-8 mechanical mobile platform (Section 2.1) from [The Machine Lab ©]. Phidgets components provide an easy to use hardware interface (Section 2.2) and software interface (Section 3.1) for controlling the robot while the main computational processing is performed by a Intel Core 2 Duo based Apple Mac Mini board (Section 2.3). Various sensors (Section 2.4) allow our robot to perceive information about its environment. A Bumblebee 2 stereo-vision camera and a Apple iSight camera module are used for semi-autonomous navigation and environment mapping. Mounted on top of a flexible pan/tilt system, controlled by powerful servo-motors, this stereo-vision camera can be moved to some extend independently of the robot platform. Sonar sensors allow the robot to avoid obstacles, visible and invisible light lasers support camera vision and voltage and ampere sensors monitor the robots battery status and runtime. User-control and software updates can be achieved by wireless communication (Section 2.5).

2.1 MMP-8 Mobile Platform

The MMP-8 (Figure 2.1) is a small platform that consists of two separated anodized aluminum housings. Inside each housing three high-torque gear-head motors are mounted, making a six wheel differential drive. Both platform parts are connected with a passive suspension joint. By default the MMP-8 is equipped with

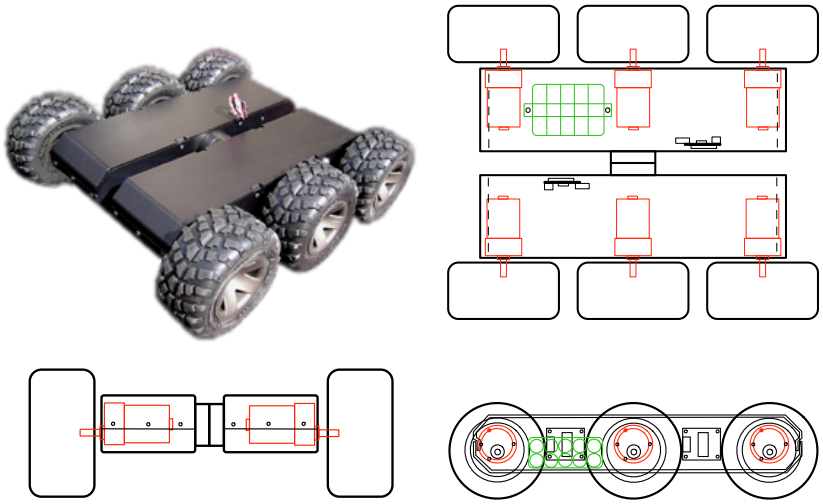


Figure 2.1: *MMP-8 Mobile Platform.*

two 10 Ampere *PWM*[1] motor controllers and a $12V$ NiMh $1400mAh$ rechargeable battery pack that is providing the system power for an runtime of over 45 minutes. As our robot has much higher power requirements, the standard $12V$ NiMh battery pack was removed and replaced by two $18.5V$ Li-Ion battery packs. Power control and distribution is accomplished by a small ATX power supply. The MMP-8 platform can carry a payload of approximately $2700g$ while weighting only $3400g$. The platform weight includes the standard battery pack, the gear-head motors and motor controllers.

2.1.1 Drive Motors

Six gear-head motors (Figure 2.2) provide the robots drive system. Gear-head motors have, as their name suggests, gears integrated to slow down their motor speed and to increase torque. The used motors have a gear ratio of $30 : 1$ so that the motor shaft rotates at a maximum of $200rpm$. As mentioned before, each housing of the robot platform contains three gear-head motors. These motors are connected in parallel (Figure 2.31) to one motor controller, so they act synchronously. This control results in differential steering.

[1] PWM stands for Pulse-Width Modulation.

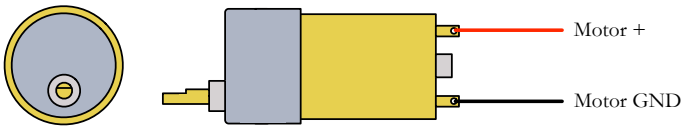


Figure 2.2: *Gear-head motor.*

Differential Steering

Differential steering is sometimes referred as ”tank-type” steering. The similarity is in the way an operator can separately control the speeds of the wheels on the left and right side to cause a directional change. Figure 2.3 illustrates how controlling the speed and direction of the wheels with differential steering can result in all types of directional motion for the robot. With differential steering no motor is required to turn any wheels to steer. Spinning around the robots center axis is accomplished by moving the left side wheels in one direction and the right side wheels in the opposite direction. A sharp turn is accomplished by solely moving the wheels of one side forward or backward. An infinite variety of turns are accomplished by moving the wheels at different speeds.
As three gear-head motors are always connected in parallel to one motor controller only a single command has to be supplied. Note the mirrored motor mounting in Figure 2.3. The motors in the platform housings are turned by $180°$ compared to each other. If the same movement direction is sent to the motor controllers it will result in a turn instead a straight forward wheel movement. The command direction has to be therefore reversed for one of the motor controllers.

Differential steering exposes in practice several problems. First of all the robot will not even drive in a straight line. This is caused by slight differences of the gear-head motors. Though they are all the same type, each motor rotates at a slight different speed while connected in parallel. This causes problems that have to be compensated by sending different speed commands to each motor controller according to a pre-calibration. Second wheel friction is an important factor. The robot will move differently according to the texture of the ground. This cannot be compensated by the robots drive system. Curves cause furthermore the front-most and back-most wheels to be dragged over the ground. Depending on the current ground texture, friction applies a force against the robots motion direction.

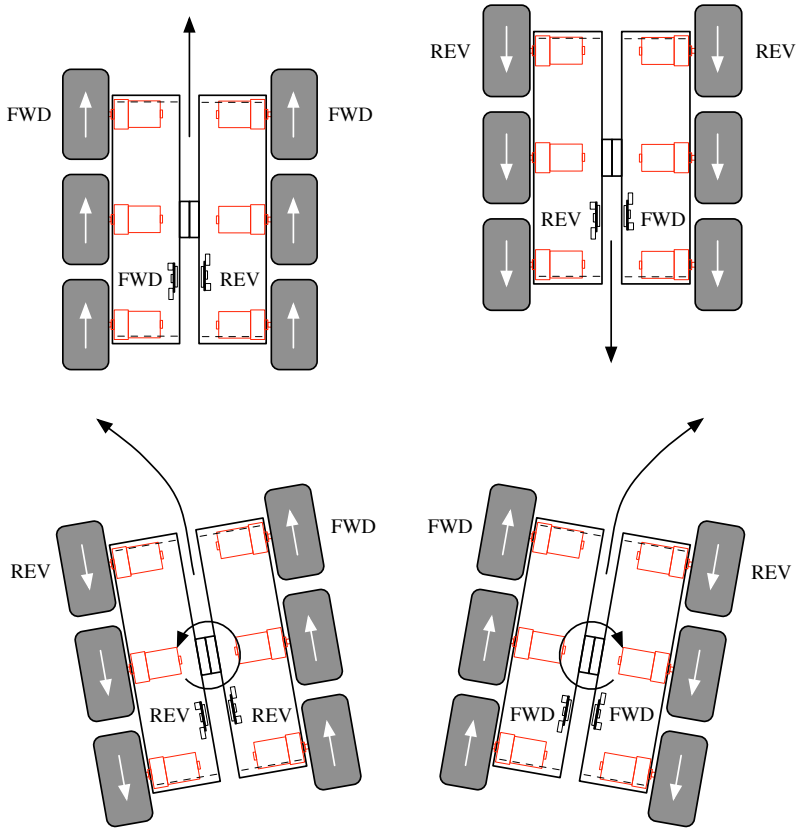


Figure 2.3: *Differential drive.*

Power Consumption

The gear-head motors are connected to the $12V$ motor bus of the motor controllers (Figure 2.31). There current draw is thereby limited by the motor controllers. Power consumption details are given in Figure 2.4. Each gear-head motor can draw a current of $3.3A$ if stalled, resulting in a maximal current draw of $10A$ per motor controller.

2.1.2 Motor Controllers

Motor controllers govern the performance of the gear-head motors. Providing a single motor channel at $10A$, the motor controller shuts down if a current of more than $12A$ is drawn. The input voltage range is between $6 - 50V$. The motor controller itself is connected to the ATX power supplies $12V$ bus forwarding this

Voltage	12 V
Stall current	3.3 A
Current at rated load	730 mA
No load current	115 mA

Figure 2.4: *Gear-head motor power consumption.*

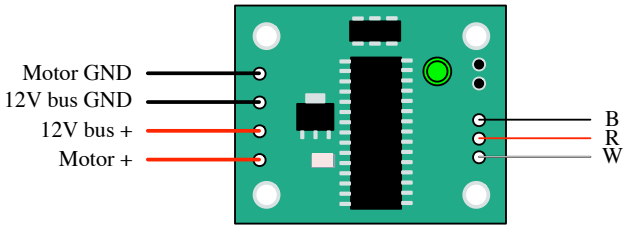


Figure 2.5: *Motor controller.*

voltage to the gear-head motors. They are controlled by Pulse-Width Modulation (*PWM*) a simple and widely used motor controller format. The motor controller's processor responds hereby to the electrical impulses send to it. The duration of these impulses (pulse-width) controls the amount and direction of current going to a motor. PWM impulses are sent from a servo motor controller over the white *W* wire (Figure 2.31). The incoming signal is 4 times averaged. No signal causes the controller to fail-safe to dead band. This is a pulse-width of $1.5ms$ and means that no current is provided to the motors. Pulses between $1.0ms$ to $2.0ms$ outside of the dead band will control the direction and the speed of the robot platform. This specific type of motor controller provides additionally $+5V$ on the red *R* wire. This wire has to be disconnected to prevent damage of the motor and the servo motor controller as the servo controller provides power on this wire too. Figure 2.6 shows a series of pulses, each with a $1ms$ pulse width, separated by a $5ms$ delay. If this pulse series is send to a motor controller, the full voltage would be applied to the attached gear-head motors resulting in a rotation with maximum speed into one direction.

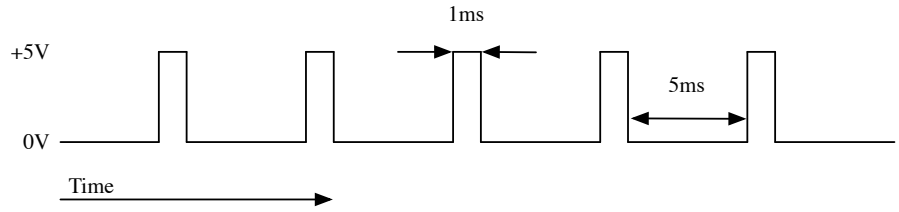


Figure 2.6: *A series of 1ms pulses over time separated by a 5ms delay.*

Calibration

Calibration or pre-calibration of the robots drive system is necessary to compensate for mechanical deficiencies. Though the dead-band is defined at $1.5ms$ its position and width have to be experimentally determined by stepwise adjusting the pulse-width. Maximum- and minimum pulses are $0.5ms$ above and below the found center position. To correct roughly for straight driving the motor speeds on both sides of the platform have to be matched on a typical operating surface. This calibration procedure has to be repeated for several movement speeds of the robot. Pre-calibrating the robots drive system results in an improved behavior but is unable to compensate for slipping wheels and alternating surfaces.

Power Consumption

The power consumption of the motor controller itself is unknown. The maximum current draw under the unlikely case that all motors are stalled would be approximately $20A$. This extensive current draw would result in an immediate shutdown of the ATX power supply as it can only provide for a very short time a maximal current of $8A$ on the $12V$ bus (Section 2.1.5). To prevent damage, this case is monitored by a current sensor.

2.1.3 Suspension Pivot

A passive suspension pivot bushing (Figure 2.7) connects the two platform housings allowing fluid motion over obstacles. Wires are routed through the aluminum tube shaft to keep everything internal.

2.1.4 Batteries

The MMP-8 robot platform is equipped with nickel metal hydride (*NiMh*) batteries. The standard pack consists of ten $1.2V$, 1400*mAh* cells. Serial connection of

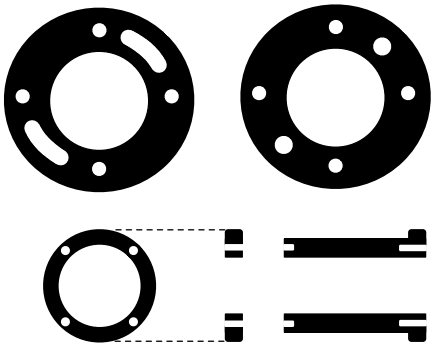


Figure 2.7: *Suspension Pivot.*

the cells results in a $12V$ nominal voltage for operation. According to the manufacturer [The Machine Lab ©] a runtime of approximately 45 minutes can be expected with the bare robot platform. As we are using several additional high power demanding components, the standard battery pack is not sufficient. *NiMh* batteries have also disadvantages, so that we choose lithium ion (*Li-Ion*) batteries instead.

Nickel Metal Hydride (*NiMH*)

Nickel metal hybrid batteries don't require complete discharging before recharging like Nickel-cadmium (*NiCad*) batteries, and they have a much higher energy density. Unfortunately, if left unused, they lose their charge faster than any other battery type (as much as 30% per month).

Lithium Ion (*Li-Ion*)

Lithium ion batteries are the standard batteries of modern laptop computers. A *Li-Ion* cell can deliver three times the energy as a comparable *NiCad* or *NiMh* battery, in a much smaller package. *Li-Ion* batteries also have a very slow self-discharge rate but are unfortunately still very expensive.

We decided to equip our robot platform with two $18.5V$ $4800mAh$ *Li-Ion* battery packs from [BatterySpace, 2009]. These packs fit perfectly between the gearhead motors in the front part of the robot chassis (Figure 2.27). Each of this battery packs is made of ten $3.6V$ $2400mAh$, 18650[2] *Li-Ion* cells. The cells are configured

[2]Li-Ion cell $2400 - 2600mAh$ is classified as not reliable by [BatterySpace, 2009] and should be replaced for future development with $3.7V$ Polymer Li-Ion cell $5000mAh$.

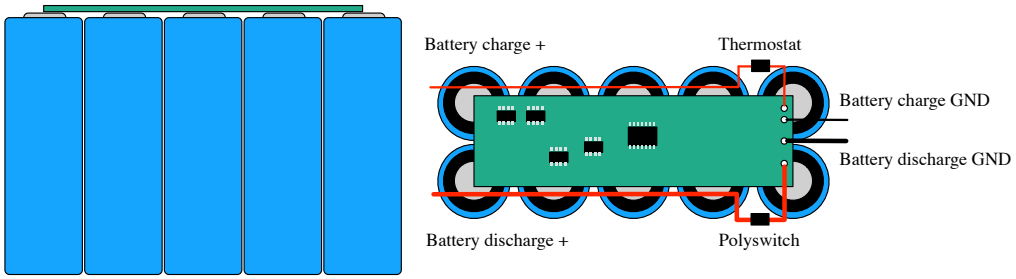


Figure 2.8: *Li-Ion battery pack.*

in a battery pack (5S/2P). This means five cells are connected in series and two five cell packs are connected in parallel. Resulting in the voltage $5 \times 3.6V = 18V$[3] and a capacity of $2 \times 2400mAh = 4800mAh$. During usage the battery pack should provide an unregulated voltage between $12.5V$ and $21V$. $21V$ is the peak voltage after completely charging the battery pack and $12.5V$ represents the completely discharged battery pack. The working voltage lies somewhere in between.

Because *Li-Ion* batteries can be dangerous if not handled properly additional protection circuits are installed for protecting the battery cells. A *PCB* protects the battery from over-charge, over-discharge, over-drain and short circuits. One *polyswitch* limits the maximum discharging current and protects from wrong polarity. Finally a *thermostat* in the charging terminal double protects the battery pack from overcharging. Figure 2.8 shows the assembled battery pack, with all components connected together. The battery pack is shrink-wrapped inside a permanent plastic covering, weighting approximately $470g$.

PCB (Protection Circuit Module)

Due to their high energy density *Li-Ion* batteries must be connected to a PCB to prevent accidental battery explosion caused by over-charging, over-discharging or over-drain. The PCB used in our battery pack has the following specifications:

- Over-charge protection voltage for single cell: $4.35V \pm 0.025V$
- Over-discharge protection voltage for single cell: $2.40V \pm 0.08V$
- Over-current detection protection: $9A \pm 1A$
- Over-charge ($> 21V$), Over-discharge ($< 12.5V$), Over-drain ($> 9A$)

[3] [BatterySpace, 2009] sells this pack as $18.5V$.

Polyswitch

Polyswitch devices are used to help protect against harmful over current surges, and over temperature faults. Like traditional fuses, these devices limit the flow of dangerously high current during fault conditions. The polyswitch device, however, resets after the fault is cleared and power to the circuit is removed. A polyswitch with a maximum current of $4.2A$ is installed for each battery pack.

Thermostat

The thermostat is basically a temperature switch. A thermostat will cut off the power when the temperature nearby reaches close to $55°C$, and is designed for protecting the battery pack from over-charging and over-discharging. When the temperature drops below the setting the power will open again. In order to protect the battery efficiently, the thermostat is placed on the side of battery pack and close to the surface of battery cell (Figure 2.8).

Charging Time

Each battery pack can be charged within approximately 115*mins* at 4800*mAh*. Charging is done with a standard $12V$ ATX power supply and a corresponding multi-charger (Multiplex LN-5014) that constantly monitors the battery charge and cuts off power when the maximum charge is achieved.

Runtime

Figure 2.9 shows a de-charge curve measured with an external voltage sensor for both battery packs in a parallel connection configuration (Figure 2.27). The runtime of the robot platform was determined to be 152*mins* under full processor load and with all components active. As the internal voltage sensor (Section 2.2.1) has shown not to be accurate enough for monitoring the battery runtime it is solely used to securely shut down the robot shortly before complete power loss. The Mid-point voltage (MPV), which represents the average operating voltage, was determined at $18.5V$.

Unregulated Voltage

The used battery pack supplies an unregulated voltage. An unregulated voltage means that the voltage drifts over time. Fully charged, the voltage is maximized at $20.1V$ and while the batteries drain the voltage will drop down to $17.4V$. See also Figure 2.9. It should be noted that our measurements don't correspond to the theoretical values outlined before.

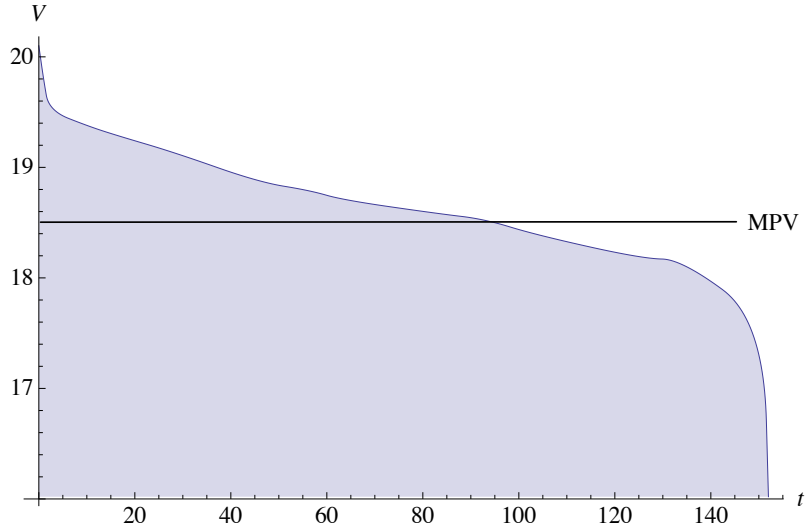


Figure 2.9: *Li-Ion battery discharge curve.*

Regulated Voltage

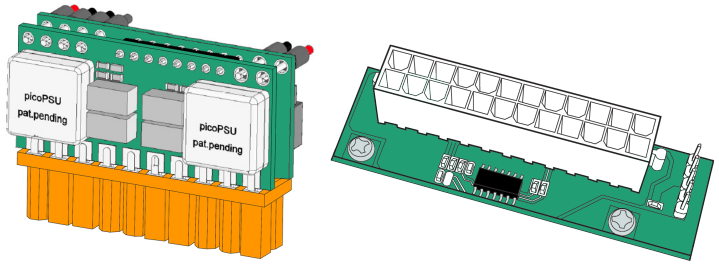
Almost all electronic devices require a constant voltage to function properly. Therefor an electronic component is needed to keep the voltage constant regardless of the battery charge and the load that is presented to it. A voltage regulator is able to create a regulated voltage from a larger unregulated voltage.

There are only two components in our robotic system that are powered by the unregulated voltage, the pico ATX power supply used for power control and distribution and the Apple Mac Mini which includes its own voltage regulators.

2.1.5 Power Control and Distribution

We use a pico sized ATX power supply for voltage regulation and power distribution. The picoPSU-120-WI-25 (Figure 2.10(a)) measures only $44.5 \times 20 \times 30mm$ fitting in a standard ATX socket that is mounted inside the robot housing (Figure 2.10(b), Figure 2.27). This specific version accepts an unregulated input voltage of $12-25V$ and provides different regulated voltage outputs at $3.3V$, $5V$ and $12V$. The maximum load for each voltage output is at $6A$, while a peak load should not exceed $8A$ for more than 60 seconds. Figure 2.28 shows the connections of the power supply in detail[4].

[4]For future development an additional ATX power supply should be considered as the current configuration reaches the specification limits.



(a) (b)

Figure 2.10: *(a) picoPSU-120-WI-25. (b) ATX PS01.*

The power state of the picoPSU can be controlled electronically via the hardware interface. This allows disconnecting components, like gear-head motors, lasers, servo-motors as well as the camera from the power bus to reduce significantly the power consumption during computational processing when the robot platform is in a fixed state. A LED on the robot housing shows the current status of the power supply.

2.2 Hardware Interface

The hardware interface is the bridge between the Mac Mini board and the robot platform. Our Mac Mini model[5] supports 4 USB ports that are suitable for connecting a hardware interface. Access and control of hardware components should be as easy as possible. We therefore decided to use *Phidgets* [Phidgets, 2009] as a low cost interface solution that is available for different operating systems and programming languages.

2.2.1 Phidgets

Phidgets are a set of components for USB sensing and control. They can be managed easily with a robust Application Program Interface (API) library. Up to 127 Phidget devices can be connected using the USB bus, making this solution suitable even for larger projects. Phidgets run usually as USB 1.1 low speed devices but are compatible to USB 2.0.

[5]Production date late 2006.

Text LCD

To be able to provide a minimal user feedback about the state of the robot platform we are using the Phidget Text LCD component (Figure 2.11). It integrates a 2-line by 20-character LCD screen that allows displaying short messages. These are in detail, status messages and possible errors of our robot.

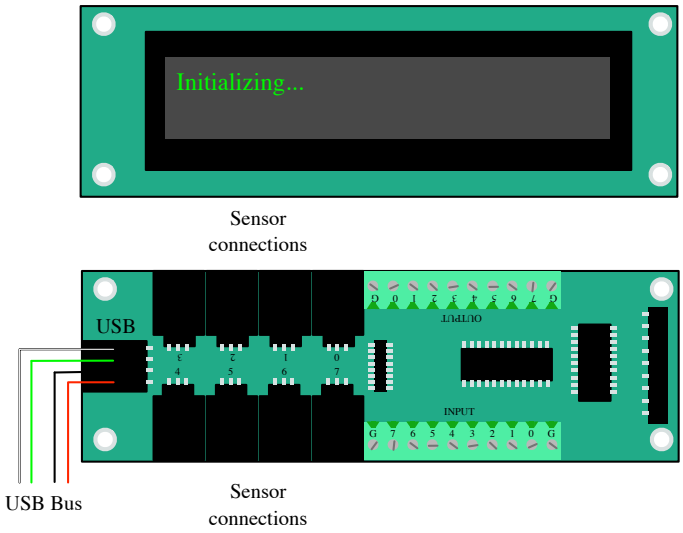


Figure 2.11: *Phidget Text LCD and Interface Kit 8/8/8.*

The Text LCD component is directly connected to the Mac Mini board (Figure 2.34), so that it is powered and active simultaneously. This is necessary as the picoPSU is controlled with a digital switch and the interface kit available on the Text LCD component.

A digital switch is always required if high current devices need to be controlled. The interface kit itself can only provide a maximum current of $15mA$ at $5V$ on the digital outputs.

Implementing a digital switch is simple. Instead directly using the interface kit for switching the state of our high power demanding device, we use the digital output to power a metal-oxide-semiconductor field-effect transistor (MOSFET) that will act as a switch to the ground for our device (Figure 2.29).

An MOSFET works similar to a semiconductor implementation of a relay. It has three leads, known as the source, the drain and the gate (Figure 2.12). Source and drain are connected with a layer of semiconductor material. The composition of the material is such that current cannot normally flow through it. The gate lead is connected to a conductive electrode that lies on top of the semiconductor

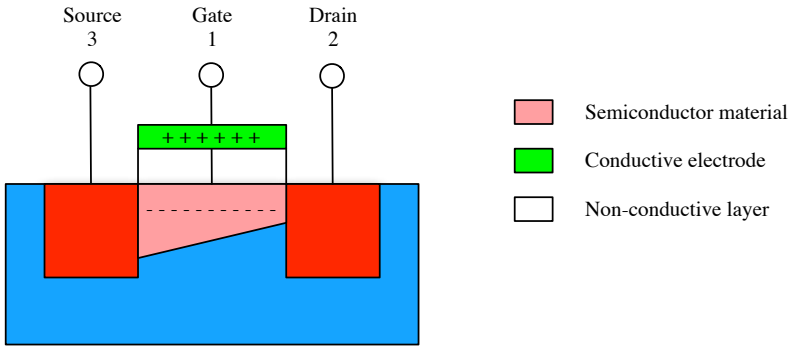


Figure 2.12: *Working principle of metal-oxide-semiconductor field-effect transistor (MOSFET).*

layer and is insulated from it by a thin non-conducting layer. When voltage is applied to the gate lead, it creates an electric field that rearranges the electrons in the underlying semiconductor layer. With the field present, current is able to flow between the source and drain. When the voltage is removed from the gate lead, the electric field reverses and current is unable to flow. The MOSFET acts as a voltage-controlled switch, where the interface kit will control the current flow between the drain and source.

Beside power control, digital switches are used in our robotic system for controlling the robots laser modules (Section 2.4.4).

Monitoring the power consumption and battery status is essential for protecting the Mac Mini board and securely shutting it down in the event of power loss. Hence Phidget Voltage and Current sensors are connected to the Text LCD component being available whenever the Mac Mini is running.

Voltage Sensor and Current Sensor

The voltage sensor[6] (Figure 2.13(a)) measures DC (direct current) voltages from $-30V$ to $+30V$ at a precision of $\pm 1V$. Given this coarse resolution it is not possible to determine the remaining battery runtime. The battery voltage change from fully charged $(20.1V)$ to completely discharged $(17.4V)$ is barely $3V$. Therefore we only monitor the crossing of the batteries de-charge voltage.

Power consumption is measured with a $20A$ current sensor (Figure 2.13(b)). Its solely purpose is to identify fault states with immense power consumption and to protect against hardware damages.

[6]Should be replaced for future development with the Phidgets Precision Voltage Sensor.

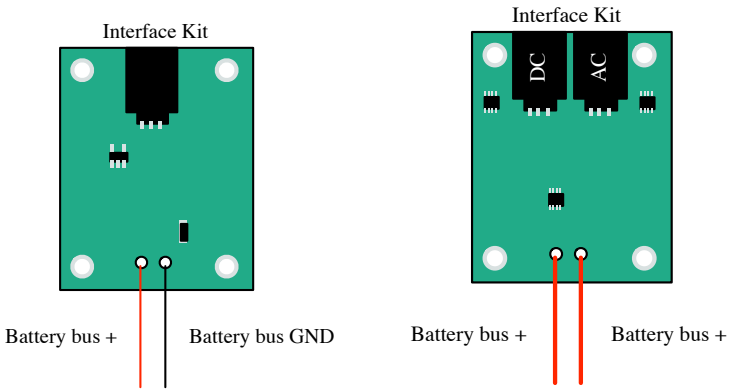


Figure 2.13: *(a) Phidget voltage sensor. (b) Phidget current sensor.*

Figure 2.33 shows the connection of both sensors to the robots batteries.

Interface Kit

The standard Phidget interface kit has 8 digital inputs, 8 digital outputs and 8 analog sensor connectors, thus it is referred as 8/8/8. In addition to the interface kit on the Text LCD component a second interface kit (Figure 2.14) is located inside the robot housing. Two line laser modules are connected with digital switches to its digital outputs, while the front sonar sensor is connected to an analog input (Figure 2.30). Using the interface kits integrated USB hub a Phidget 4 Motor servo-controller is linked.

Phidget 4/1 Motor Servo-Controller

Motor servo-controllers are used to generate PWM impulses by turning rapidly on and off the voltage for a very specific amount of time. The longer the voltage is turned on, the larger the pulse width will be. Servo-motors (Section 2.2.3) have a built in electronics that allows them to directly respond to different pulse widths. Gear-head motors are missing this electronics so they need to be controlled indirectly over motor controllers, which first translate the pulse width to a specific current (Section 2.1.2).

Our system uses two motor servo-controllers (Figures 2.15), to control 4 respectively 1 motor. Figures 2.31 and Figure 2.32 depict the servo motor connections of the 4 Motor Servo-Controller. This servo-controller governs the pan/tilt system for the stereo-vision camera, the laser deflection and one section of the

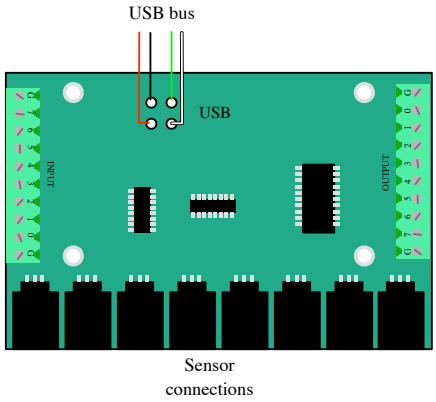


Figure 2.14: *Phidget Interface Kit 8/8/8.*

gear-head motors. The 1 Motor Servo-Controller governs the remaining gear-head motors (Figure 2.31).

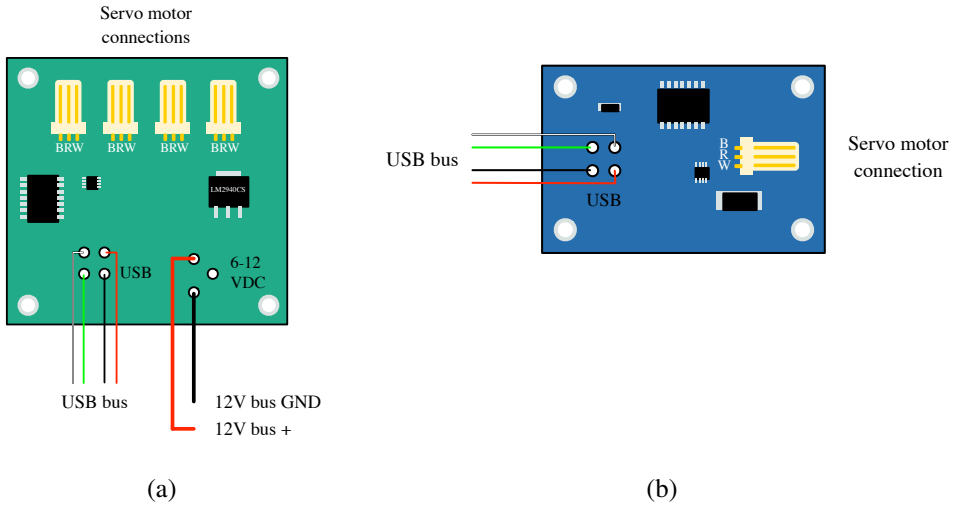
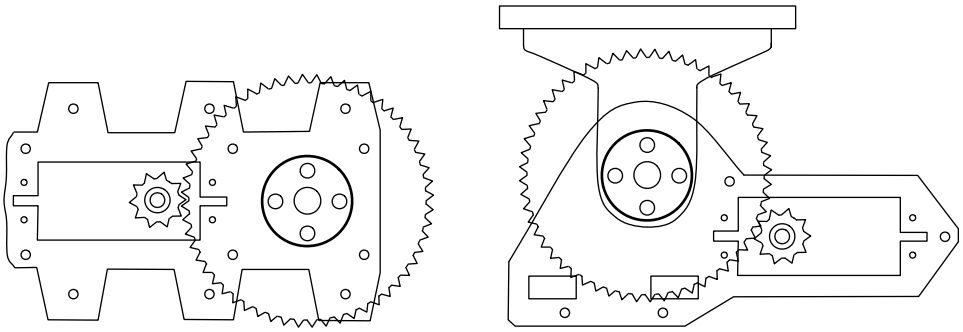


Figure 2.15: *(a) Phidget 4-motor and (b) Phidget 1-motor servo controller.*

2.2.2 Pan-Tilt System

The stereo-vision camera can be moved to some extend independent of the robot platform by utilizing a pan/tilt system (Figure 2.16). Directly attaching the camera

to the servo motors showed to be inappropriate as no smooth control and movement repeatability could be achieved. This was mainly due to the camera's weight causing a decreased servo motor precision. Hence we decided to use the ServoCity [ServoCity, 2009] SPG400 Gear Drive System for pan and the ServoCity SPT400 Gear Drive System for tilt control.



(a) (b)

Figure 2.16: *(a) ServoCity SPG400* $5:1$ *Gear Drive Pan System and (b) ServoCity SPT400* $5:1$ *Gear DriveTilt System.*

Both systems can use any standard size servos (Section 2.2.3). As the gear ratio we choose $5:1$, which increases the stated power rating and transit time of our servos to 5 times. By bypassing the servos internal potentiometer and utilizing an external precision potentiometer, the amount of operating rotation stays the same and precision increases. By not transmitting the servo power through the shaft but directly to the gear hub ensures that no slipping occurs.

2.2.3 Servos

Servos (Figure 2.17) are electro-mechanical devices most commonly found in radio controlled (R/C) devices. Their sole purpose is to rotate a tiny shaft extending from the top of the servo housing. Extending from the side of the servo is a thin cable comprised of three wires. Two wires are used to send power to the servos motor and one wire is used to send commands to the servo. Each servo has a built-in processor that responds to PWM impulses sent to it. The servo interprets the pulse from the servo controller and rotates its shaft either clockwise or counterclockwise based on the pulse widths. Most servos have a pulse width limit between $1.0ms$ and $2.0ms$. A pulse width in between is interpreted as positions, while the center position (neutral) is at $1.5ms$. When the servo detects a well-defined pulse, the servo shaft will begin rotating toward the position that

corresponds to that pulse width. It requires several pulses for the servo to reach that position.

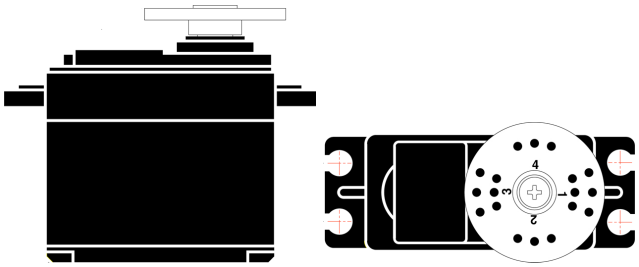


Figure 2.17: *Standard servo motor.*

We are using three different servo motors as listed in Figure 2.18. Please refer also to Figure 2.32 for the servo motor connection plan.

Servo type	Control
Hitec Digital HS-5645MG	Pan
Hitec Standard HS-645MG	Tilt
Hitec Standard HS-475HB	Laser

Figure 2.18: *Hitec Servos.*

The digital high torque servo HS-5646MG is controlling the cameras rotation around the Y-axis (pan). A digital servo is employed because it has many more rotation steps compared to its conventional analog version. This means the servo is capable of finer adjustments. The same high torque servo HS-645MG, but in the analog version, rotates the camera around the X-axis (tilt). Precision for tilt control is less relevant but a similar torque for handling the cameras weight is needed. Finally a standard servo HS-475HB controls the laser deflection system.

2.3 Computational Processing

Robots require some kind of processing unit that is able to control the hardware in an intelligent way. Sensor information need to be read over the hardware interface, processed and analyzed. A user programmed routine reacts to the results of this analysis and sends commands back to the hardware interface for a controlled

robot behavior. The continuous execution of these steps results in an autonomous robotic system.

Computational processing has to be performed in real-time. As we are using a stereo-vision camera as a sensor, this includes the processing and analysis of high-resolution stereo images. The processing power we can employ is limited by the fact that we need a light and small unit that can be fixed on the robotic platform. Additionally low-power consumption is required as the robot is powered by batteries. For this reasons we choose the Apple Intel Mac Mini as the core unit for computational processing. It represents for us the best compromise between size, weight, power-consumption and processing power.

2.3.1 Apple Mac Mini

Our Apple Mac Mini model (Figure 2.19) features a $2GHz$ Core 2 Duo processor with $2GB$ SDRAM ($667Mhz$), a $120GB$ hard disk ($5700rpm$) and a built in GMA 950 graphics processor with $64MB$ shared memory. Built-in $54Mbit/s$ Wi-Fi (802.1g) and Bluetooth 2.0+EDR (Enhanced Data Rate) allow the wireless communication. One firewire port ($400Mbit/s$) is connected to the Bumblebee 2 stereo-vision camera, one USB port ($480Mbit/s$) is connected to the iSight camera module and the remaining 3 USB ports are used for connecting the Phidget hardware interface components.

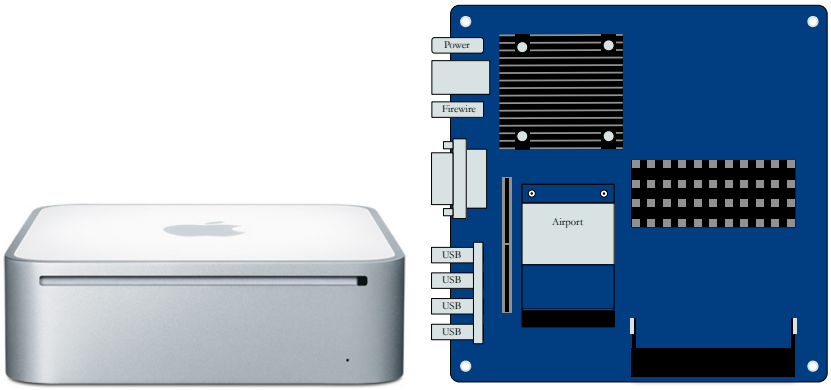


Figure 2.19: *Apple Mac Mini Intel Core 2 Duo, Model Late 2006.*

Intel's Core 2 Duo processor architecture allows the simultaneous execution of 2 threads which can be fully exploited for vision-based processing. The integrated graphics processor doesn't support hardware shaders, thus all software (Section 3) is running on the CPU.

Only the bare Mac Mini board is mounted on the robot platform. The original enclosure, the combo drive and the audio adapter have been removed, as they are not needed.

Power Requirements

The Mac Mini has integrated voltage converters tolerating an operating voltage between $12-20V$. Voltages above $20V$ lead to an increased fan activity because the voltage converters substantially increase their surface temperature. In our robot system the unregulated voltage reaches $20.1V$ with fully charged batteries. This is actually not a problem, as under load the voltage drops within seconds under the critical value. In case the voltage reduces to a value below $18.5V$, damage or fault of firewire devices can occur. This is the case after 100*mins* of runtime. For this reason we don't power the Bumblebee 2 stereo-vision camera from the firewire bus, instead it is directly connected to the $12V$ bus. The unregulated voltage stays always above $17V$ and never exceeds the lower input voltage limit.

Power Consumption

During normal use the Mac Mini typically draws $20-35W$. Under the maximal CPU load power consumption increases to $23-110W$. This corresponds to $1.24-6A$ at $18.5V$. In standby, power consumption is reduced to $11-18W$[7].

iSense

The Mac Mini logic board is not intended to be powered by batteries. An easy connection is prevented by the Apple specific power plug and a feature called iSense that is integrated into the PSU (Power Supply Unit). iSense should provide over-current, undercurrent, over-voltage and under-voltage protection. It should as well provide a feedback loop between PSU and logic board so that the PSU can regulate its output. But it is not known whether these features are actually implemented or not. To circumvent iSense and to imitate a standard power supply, we are using a $2k7\Omega$ resistor that is connected to ground. See Figure 2.20 for cable wiring details.

[7] http://support.apple.com/kb/HT3468?viewlocale=en_US.

Pin	Usage	Cable
1	GND	
2	+18.5 V	
3	iSense	
4	+18.5 V	
5	GND	

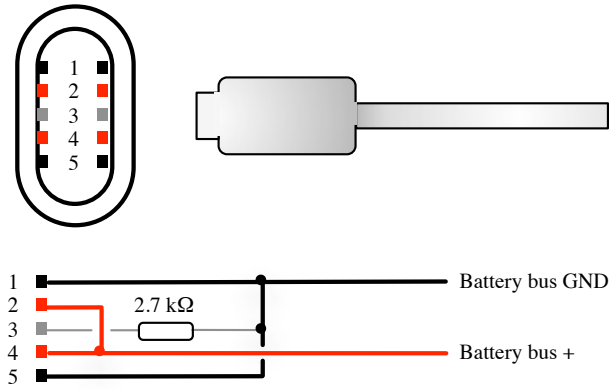


Figure 2.20: *Mac Mini power cable wiring.*

2.4 Sensors

2.4.1 Bumblebee 2 Stereo-Vision Camera

The Bumblebee 2 is a stereo-vision camera from Point Grey Research (Figure 2.21). It consists of two $1/3$" color progressive scan CCDs from Sony with a resolution of 1032×776. The maximal frame rate is limited by the firewire port to 20*fps* at a resolution of 1024×768. Besides the firewire port a GPIO (General Purpose I/O) connector is installed. The temperature near the imaging sensor can be reported through the cameras temperature sensor.

Color conversion and basic image processing can be performed on-camera. This includes the conversion to YUV411, YUV422 and RGB formats as well as the control of sharpness, hue, saturation and gamma.

The Bumblebee 2 is pre-calibrated for lens distortions and camera misalignments. Left and right images are aligned within $0.05mm$ pixel RMS (Root Mean

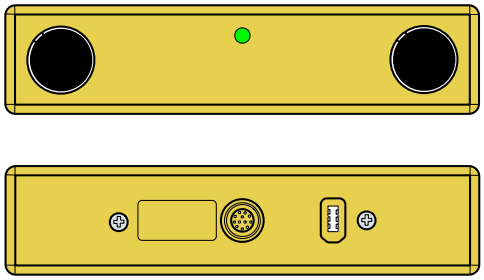


Figure 2.21: *Bumblebee 2 stereo-vison camera, Point Grey Research.*

Square) error[8]. The calibration retention system prevents the camera unit from losing calibration when the device is subject to mechanical shock and vibration. Calibration files are embedded in the camera, allowing the software to retrieve image correction information. Unfortunately the included SDK is limited to Windows and Linux operating systems. Therefore calibration and stereo-vision functionality has to be re-implemented for Mac OS X.

With $342g$ the camera adds significant weight to our robot platform, requiring a powerful pan/tilt system as described in section 2.2.2.

Stereo Image Format

The Bumblebee 2 supports the firewire transfer mode Format_7. In this mode images from both CCD sensors are sent at the same time as pixel (byte) interleaved stereo pairs. Pixel interleaved images use a raw $16bit$ pixel format, where the first byte is from the left camera and the second from the right. Color is achieved by interpolating the raw pixel values. Time information can be additionally embedded in the first pixels.

Firewire and GPIO Connector

A standard 6-pin firewire (IEEE1394) connector is used for data transmission and camera control. As described in section 2.3.1, a drop in the unregulated voltage below $18.5V$ can damage firewire devices. We therefore connect only the 4-wires responsible for data transmission, and supply the power to the camera separate over the GPIO connector. The detailed connections can be seen in Figure 2.22.

[8]Specified from Point Grey Research for a resolution of 320×240.

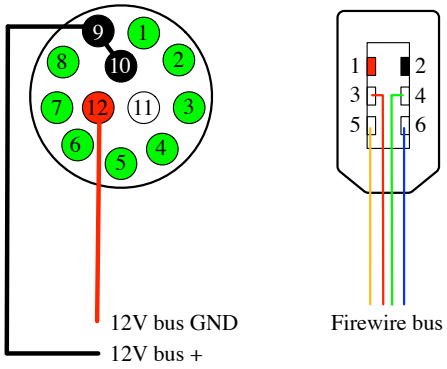


Figure 2.22: *Bumblebee 2 stereo camera connection.*

Power Consumption

The camera can be powered by an external input voltage from $8 - 32V$. Power consumption is specified at $3W$. This corresponds to $250mA$ when connected to the $12V$ bus.

2.4.2 Apple iSight Camera Module

The Apple iSight camera module (Figure 2.23) is the standard camera built into Apples Mac Book's. This module is freely available as a replacement part at low-cost. It measures only $55mm \times 5mm$ and consists of a USB 2.0 interface and camera with a CMOS active pixel sensor. Though the camera has a plastic, fixed-focus lens it acquires high quality images with a native resolution of 1024×768 at $30fps$[9]. The color format of the camera is limited to YUV422. Sharpness, hue, saturation and gamma can be controlled.

As a consumer camera the Apple iSight is not pre-calibrated. Calibration has to be done to determine the camera's intrinsic parameters. Lens distortions are corrected in software with a lookup table.

Figure 2.23: *Apple iSight camera module.*

[9] Apple camera module 820-1929-B

Image Format

Images are transferred over the USB interface in the YUV2 format using $16bit$ per-pixel. The byte order is YUYV. Color information needs to be interpolated from two consecutive pixels.

Power Consumption

Power is provided to the camera over the USB interface. The maximal power consumption reported is $100mA$ at $5V$.

2.4.3 Sonar

To aid the robots vision-based navigation and to protect the robot platform from damage, distance sensors for the recognition of obstacles are used. The most common type of distance sensor is the sonar. A sonar sensor uses the speed of sound to measure the distance to objects. Inaudible sound waves are projected out from a transmitter on the robot, bounce off of surfaces, and return to a receiver on the robot. The time it takes for the sound waves to return to the sonar sensor is used to calculate the robot's distance from an object. This data is then passed over the hardware interface to the processing unit of the robot resulting in an appropriate behavior.

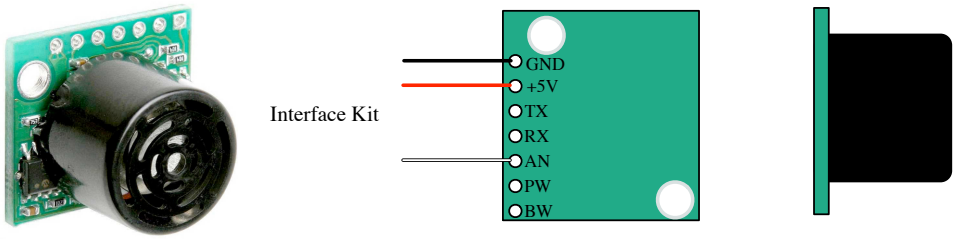


Figure 2.24: *MaxSonar EZ1 sonar sensor.*

Two MaxSonar EZ1 sonar sensors (Figure 2.24) are installed on our robot platform. Mounted on the robots housing and facing in opposite directions they cover the area in front and behind the robot. Each sensor detects objects in the range of $15cm - 6.45m$. Objects that are closer than $15cm$ are ranged at $15cm$. The sonar detection cone angel is varying according to the object distance. Figure 2.25 below shows the target detection angles of the MaxSonar EZ1. The system gain is actively and continuously adjusted by the sonar's system software to yield a long comparatively narrow beam. Hence most objects are detected in the central

$36°$ zone. The actual detection zone is broader and reaches $50°$ if there is no object within the center zone.

Having several sonar sensors active simultaneously causes a problem called cross-firing. Cross-firing leads to wrong distance estimations as sound waves from a different transmitter reach the receiver.

Synchronizing the transmitters and receivers solves this problem. Nevertheless cross-firing is quite reduced in our system as the sonar's are facing opposite directions, so for simplicity we renounce to synchronization. Multiple reflections can have the same effect as cross-firing thus we prefer to filter the sonar values in software.

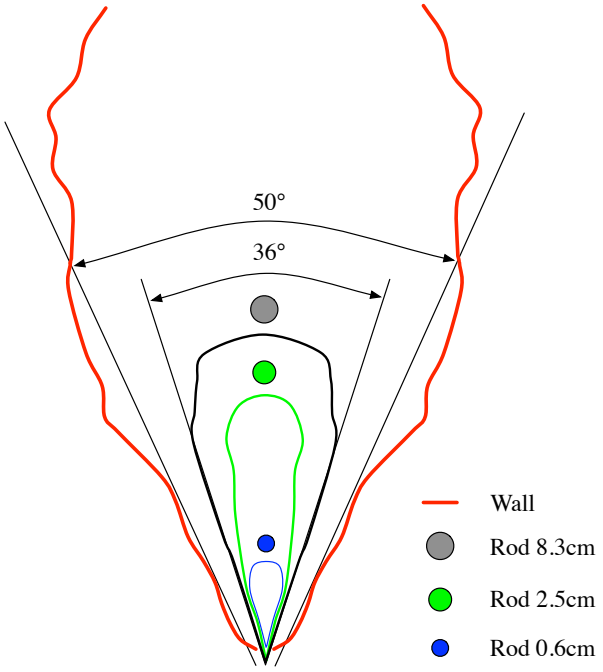


Figure 2.25: *MaxSonar EZ1 sonar detection cone angle.*

2.4.4 Visible and Invisible Light Lasers

Under unfortunate conditions the camera and the sonar sensors don't provide sufficient and precise information about the position and shape of obstacles or the environment. For this reason they are supported by line laser modules.

Red Line Laser Modul ($650nm$)

Obstacles are typically detected within a $36°$ or even $50°$ cone by the robot's sonar. The exact position of the obstacle inside the cone can't be determined, making precise navigation difficult. Hence the sonar is supported by a red line laser module. This module projects a red laser line with a wavelength of $650nm$. At a distance of 1m the line is $0.5mm$ wide. The lens for generating the laser line has a $90°$ fan angle. Despite the $5mW$ diode power the laser module is classified as Laser *Class 2M* (Section A.1), mainly due to the attenuating characteristics of the optics for generating the laser line.

We mounted the laser in the lower part of the robot housing so it is well below eye height and indicated it with a warning label. Should the sonar sensor detect an obstacle, the laser will project a line in front of the robot. The aberration of the projection from a line is used by the camera to determine the exact shape and location of an obstacle (See Section 5.1 for details).

Green Line Laser Module ($532nm$)

The Bumblebee 2 camera's stereo-vision fails in most natural indoor environments as they lack sufficient textured areas. To allow nevertheless depth estimation in such environments we are using a green line laser module with a wavelength of $532nm$. The projected line by such a laser is much better visible on long distances and in addition the camera is more perceptive in the green spectrum. As true green laser modules are still not available to public, we are using instead a DPSS (Diode-pumped solid-state) laser. This uses a powerful $200mW$, $808nm$ wavelength infrared laser diode that pumps a neodymium doped yttrium orthvanadate (Nd:YVO4) crystal, producing $1064nm$ wavelength light. The frequency is doubled using a nonlinear optical process in a KTP crystal, producing $532nm$ light resulting in a green laser with a final output of $5mW$. These characteristics qualify the laser as *Class 3R*. The line generating optics and the mirror deflection system limits (Figure 2.26) the output power, so the resulting laser beam can be considered harmless for the eye.

With the laser deflection mechanism we can control the laser line projection within an area of $60°$ in front of our robot platform and generate artificially texture for the stereo-vision camera.

Infrared Line Laser Modul ($780nm$)

As an alternative for the red line laser module for obstacle detection we considered an infrared line laser module with a wavelength of $780nm$. The advantage of operating in the invisible light range is the easier extraction of the laser line from

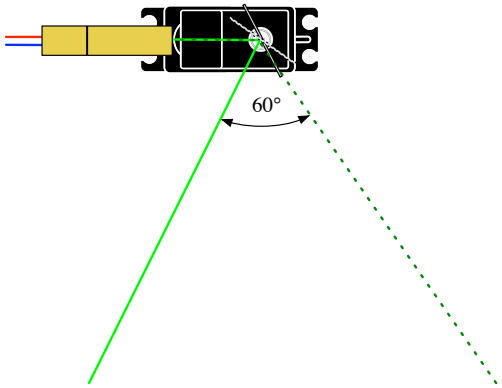


Figure 2.26: *Laser deflection mechanism.*

the image. The cameras sensor is still receptive in the infrared spectrum if the IR-filter is removed from the cameras lens system. Unfortunately this laser belongs already to laser *Class 3B*. Therefore we adopted not to use this laser and preferred the red line laser module instead.

2.5 Communication

The Mac Mini is equipped with an Airport Extreme 54*Mbit/s* card for wireless connectivity. To extend the robots operating range an external antenna is installed on the robot housing. Wireless connectivity is needed for updating the system software or remotely controlling the robot bypassing the robots computational processing unit. Being remotely connected the robots hardware interface can be accessed directly from any machine for testing and debugging. It behaves just like it would be connected to the local machine. Data transfers are limited in this operating mode. In particular images from the stereo-vision camera can only be transferred compressed and with a quarter of the original resolution before the available transfer bandwidth is exceeded. Thus this mode is inappropriate for testing vision-based navigation. The wireless connectivity provides mainly an easy access to the robotic system through Apple Remote Desktop. Programs can be transferred, debugged and run within the Xcode environment.

Though bluetooth and infrared are available on the Mac Mini they are currently not supported for remote control.

2.6 Connection plans

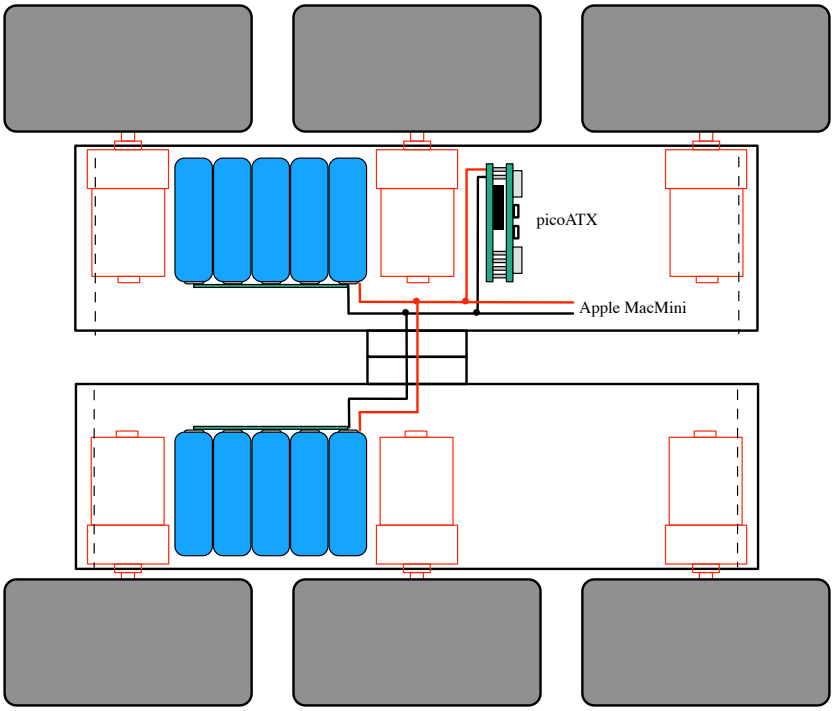


Figure 2.27: *Battery connection.*

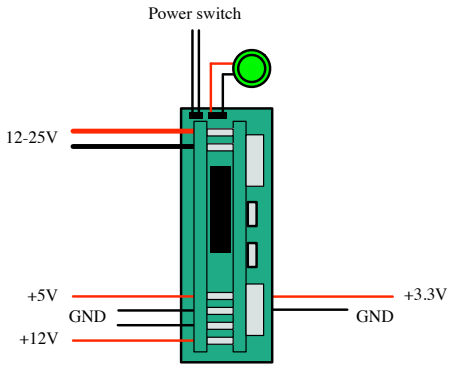


Figure 2.28: *Power distribution.*

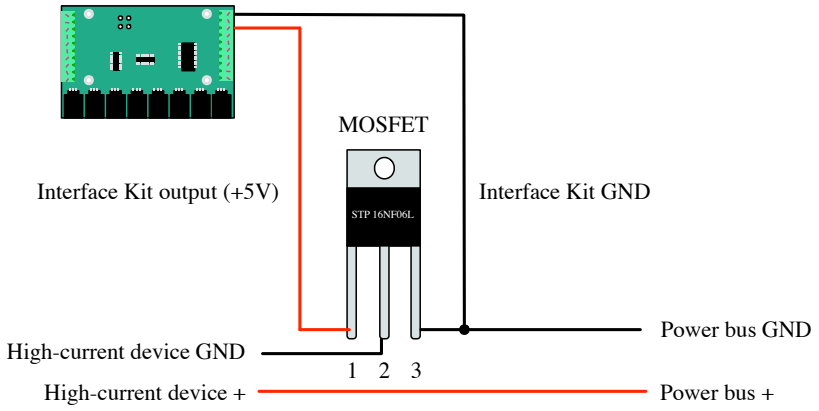


Figure 2.29: *MOSFET - switch.*

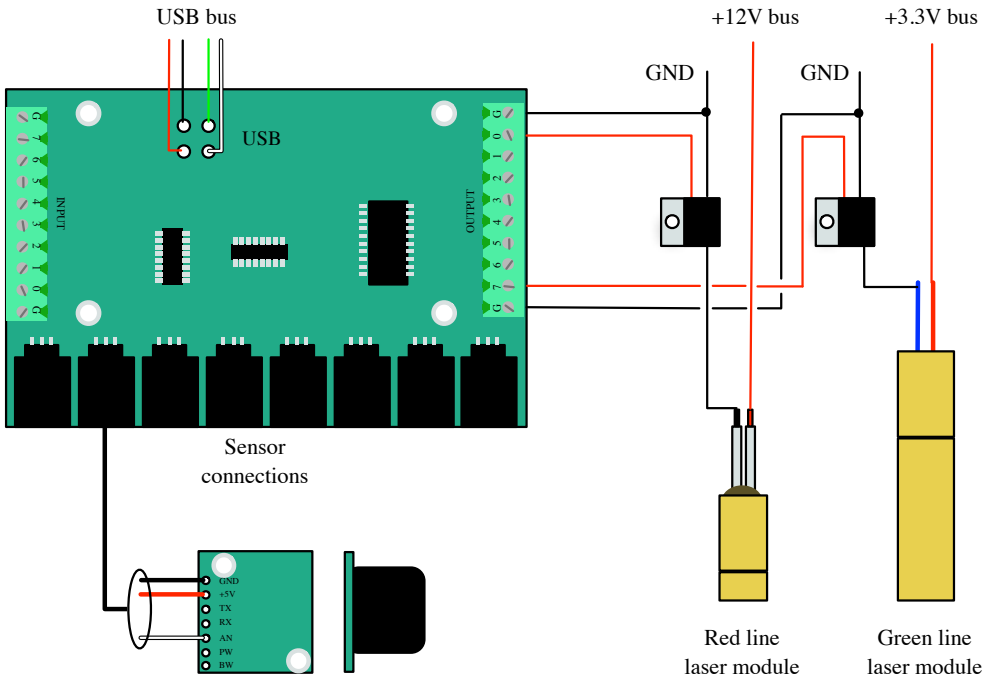


Figure 2.30: *Internal Phidget Interface Kit 8/8/8 connections.*

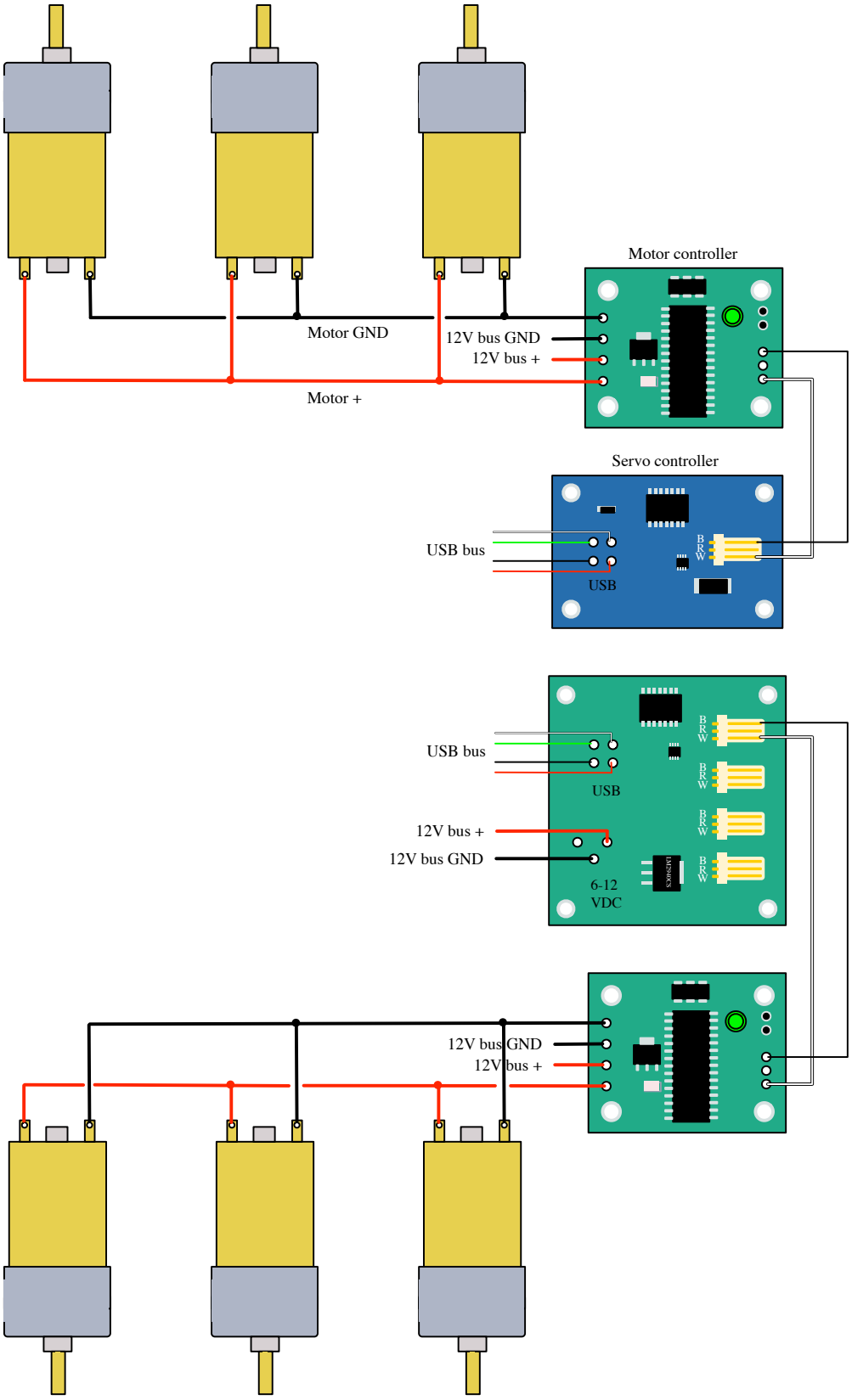


Figure 2.31: *Gear-head motor control connection.*

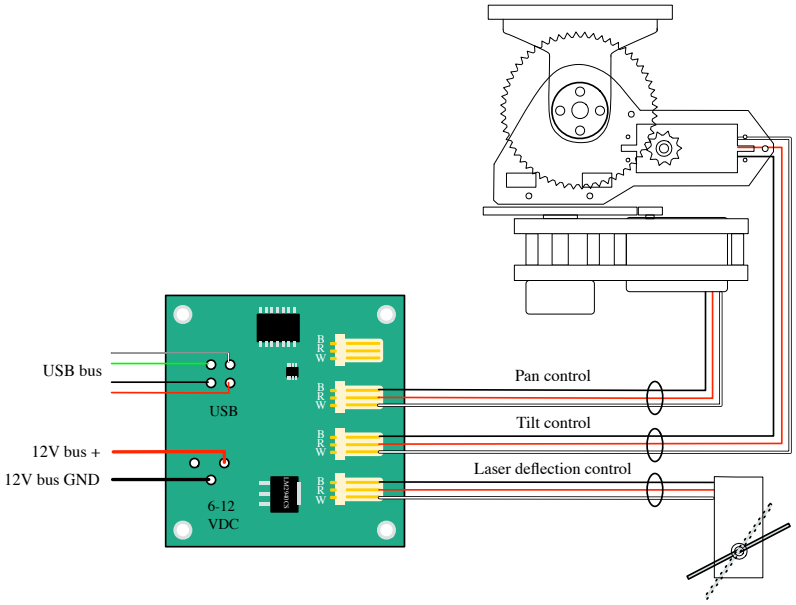


Figure 2.32: *Servo motor control connection.*

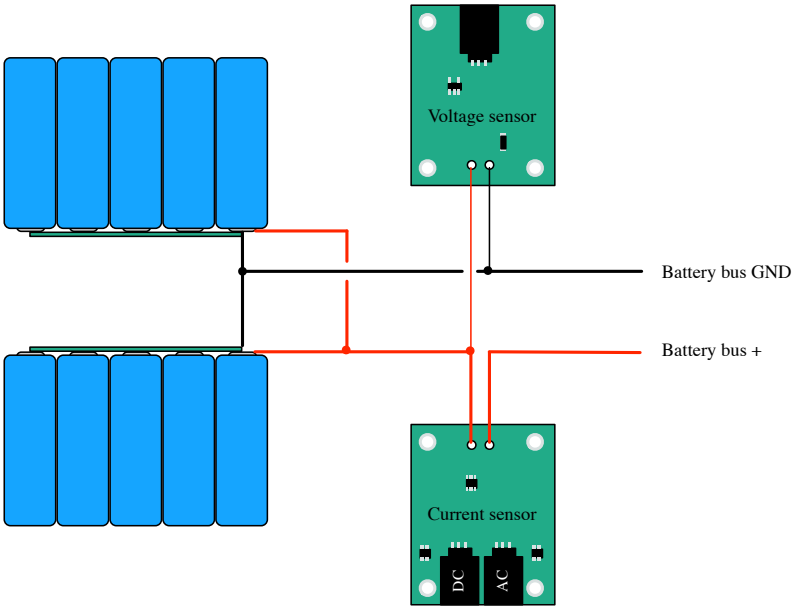


Figure 2.33: *Voltage and current sensor connected to the battery.*

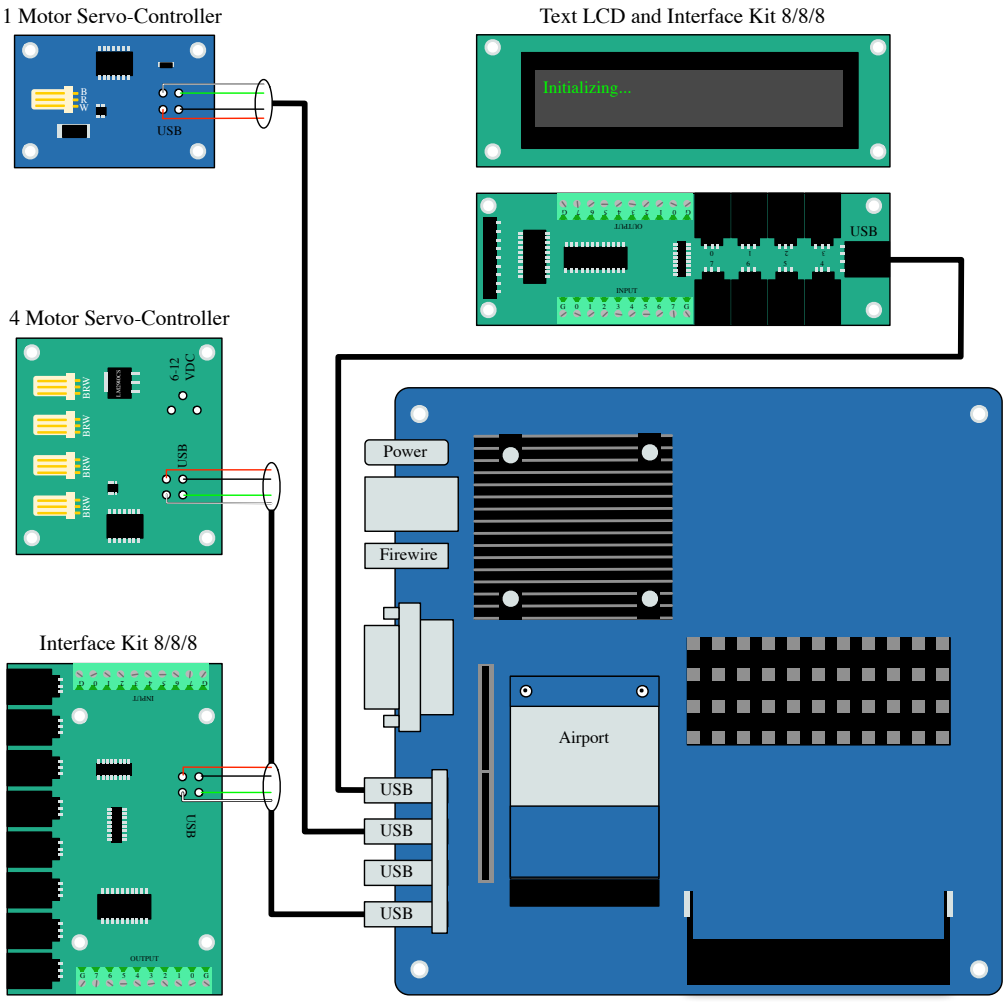


Figure 2.34: *Hardware interface USB connections.*

CHAPTER 3

ROBOT SOFTWARE

Robot software enables our mobile robot platform to perform different tasks, from interaction with the environment to semi-autonomous navigation and mapping. Varying software systems and frameworks have been proposed to make robot programming easier [Microsoft, 2009; Laboratory, 2009; MAPIR, 2009]. However we developed our own customized framework, to suite the hardware and programming needs of a real-time vision-based system.

3.1 Platform API

Our Mobile Robot Platform API (Figure 3.1) builds upon frameworks and services offered by the operating system and third parties.

Core Services provide access to system functions and Apple system events, allowing full control over the operating system. OpenGL (Open Graphics Library)[1] and OpenCV (Open Computer Vision Library) perform the on platform visualization. Visualization is limited to local debugging, as the wireless bandwidth is insufficient to transfer content in real-time and high-quality. Alternatively data acquired with the mobile robot platform can be transferred for visualization to a separate workstation.

The Phidgets and the Libdc1394 framework are of major importance for our Mobile Robot Platform API as they provide direct access to the robot hardware.

[1] http://www.opengl.org

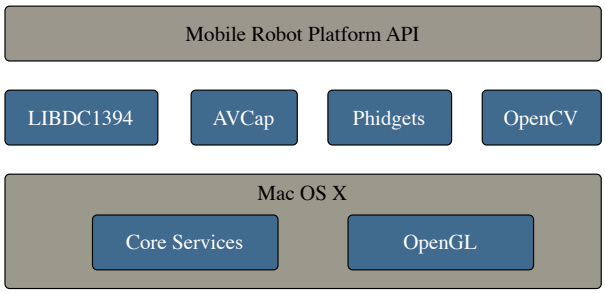


Figure 3.1: *Framework base of Mobile Robot Platform API.*

3.1.1 Libdc1394

Libdc1394 is an open source library [Source, 2009] that provides a high level programming interface to control IEEE1394 based cameras. It interfaces with the BumbleBee 2 stereo-vision camera that follows the IEEE1394 communication standard, providing a continuous stream of uncompressed images. Moreover, full access to camera registers is possible. Key registers contain the camera calibration data, adjust the on-camera image processing and initialize the stereo image format (Section 2.4.1).

Camera calibration data is utilized to generate a resolution dependent lookup table for correction of camera lens distortions. Before capturing, camera registers for brightness, gain and white balance are set to initial, environment dependent values. The camera is initialized to transmit byte interleaved stereo images at a resolution of 1024×768 with $20fps$. Libdc1394 employs a DMA ring buffer for captured frames as shown in Figure 3.2(a). A ring buffer has a certain limited capacity. In the event of an overflow, the least recent image is overwritten and the captured frame lost.

We implemented a thread based polling routine for frame handling. A capture thread is getting active, as soon as there is a new frame in the ring buffer available. The raw stereo frame is de-queued from the ring buffer, pre-processed and send to our own FIFO inbound queue.

Preprocessing de-interlaces the $16bit$ byte interleaved stereo frame to two $8bit$ sub-frames. These are decoded in parallel from their raw bayer pattern format [Ramanath et al., 2002] to planar RGB-frames and finally to $8bit$ grayscale (Figure 3.2(b)). One sub-frame is stored for later usage by the difference filter. All captured sub-frames are run through this difference filter to limit the number of frames to be processed. For this purpose the filter subtracts two consecutive sub-frames and calculates the mean value of the resulting difference frame. If the mean value

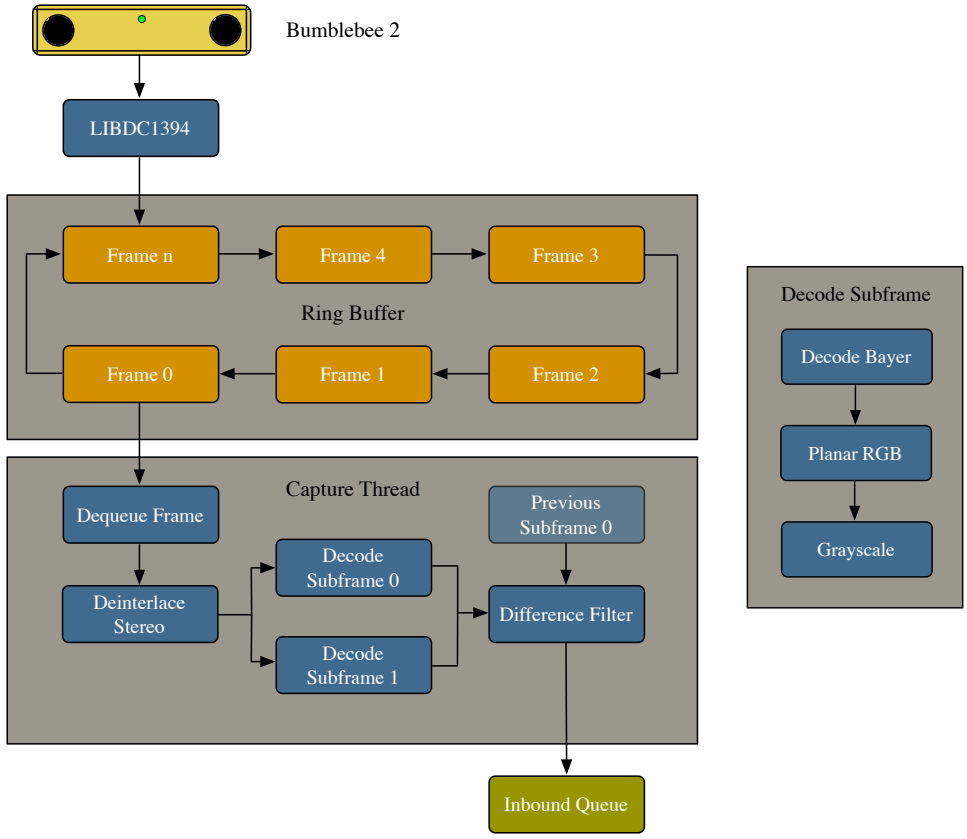


Figure 3.2: *(a) Image capturing with Libdc1394. (b) Thread to decode sub-frame.*

is below a certain threshold, both sub-frames are discarded. Image noise and brightness variations define the threshold for this filter. Ideally it is set in a way that only frames with significant changes pass the difference filter and are sent to the inbound queue. Subsequently, the frame rate varies between $0 - 20 fps$ depending to a large extent on the robot and camera motion.

The whole pre-processing is multi-threaded so that frames and sub-frames are processed in parallel. Use of SSE (Streaming SIMD Extensions) ensures that pre-processing consumes a minimal amount of available computational time and no frames from the ring buffer are lost.

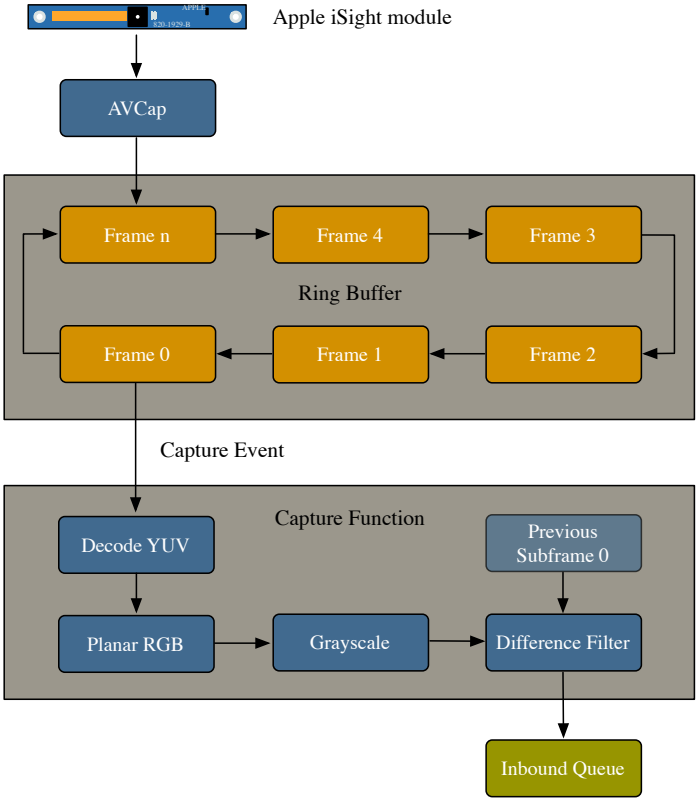


Figure 3.3: *Image capturing with AVCap.*

3.1.2 AVCap

Avcap [Source, 2010] is a cross-platform video capture library. It provides a simple and easy to use C++ interface to access USB camera devices. This library is essential to acquire images in combination with the Apple iSight camera module.

Similar to Libdc1394, the AVCap library employs a ring buffer to store captured frames. Each time a frame becomes available the user provided capture function is called (Figure 3.3). Processing time in the capture function is limited through the frame rate. The Apple iSight camera module transmits frames with a resolution of 1024×768 at $30fps$. Hence, a single frame has to be processed within one thirty of a second.

We decode the raw frame from YUV422 to planar RGB-frames and to $8bit$ grayscale. A difference filter ensures that only frames with a certain change, ideally occurring from the motion of the robot platform, are sent to our FIFO inbound queue.

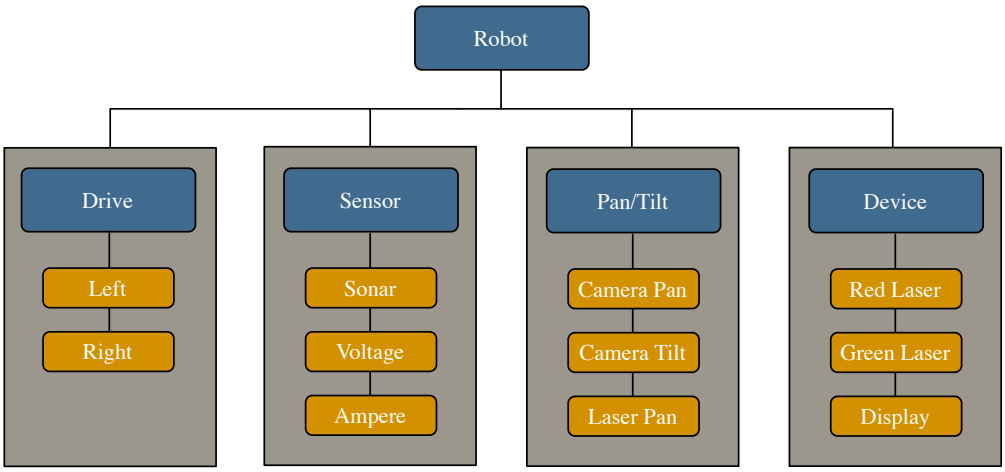


Figure 3.4: *High-level abstraction of robot hardware.*

3.1.3 Phidgets

The Phidgets framework consists of a C-library, which implements the low level protocols necessary to communicate with the hardware interface (Section 2.2). Built upon this low level library are higher level libraries that simplify the usage. We abstracted the Phidgets library further and represent the robot hardware with a specific class (Figure 3.4). Through this class it is possible to control all components in an easy and intuitive way. It should be noted, that all libraries extensively employ threading and event handling.

Phidgets provides in addition to the framework a web service, which allows the hardware interface to be controlled over any TCP/IP network. As this service works on a low level, the robotic software could be run on any computer inside the network, as if the hardware interface would be directly connected.

3.1.4 Visual Processing Pipeline

Our mobile robot system relies almost solely on vision for navigation and environment mapping. Thus we need a high-performance system to process the incoming image data at a rate of at least 30 MPixels/second[2].

We choose a multi-threaded pipeline approach for high level image processing and analysis (Figure 3.5(a)). This approach is most suited for our problem, as the same processing and analysis has to be repeated on a continuous stream of

[2]Bumblebee 2 stereo acquisition rate: $1024 \times 768 \times 20 fps \times 16 bit$

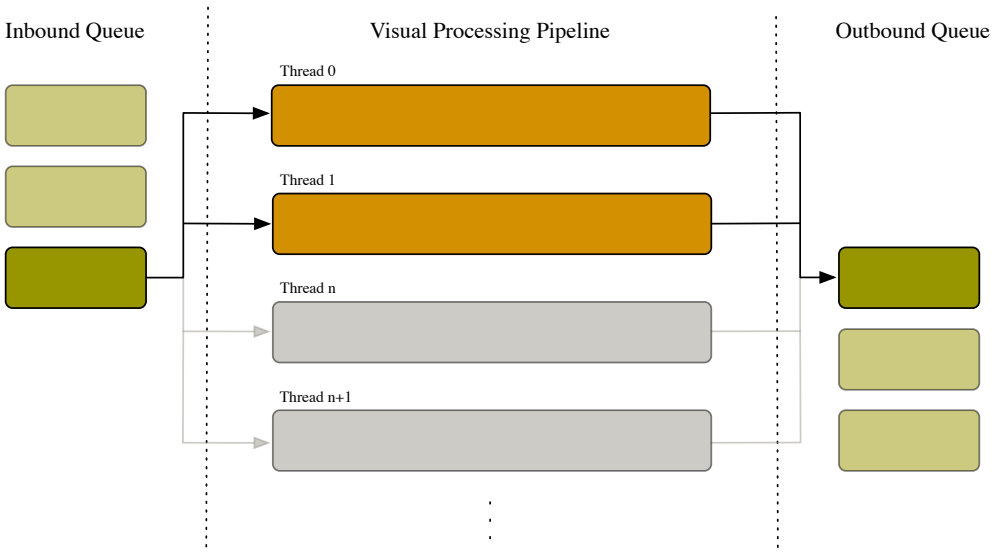
images, which is predestinated for parallelization and scales nearly perfectly linear with the number of available CPU cores in our system. The number of visual processing pipelines can be adapted to the robots computational processing unit. Experiments showed that the best ratio between pipelines and cores is $2:1$.

We can load different sequential image processing and analysis tasks into the visual processing pipeline gaining a maximal speed necessary for a real-time system. Though being sequential, independent segments are run again in parallel as we process always a pair of sub-frames as shown in Figure 3.5(b). In this specific example a visual pipeline task, stereo sub-frames are first undistorted in parallel. Brightness differences are normalized in a sequential segment, while feature detection is again independent and therefore parallelized.

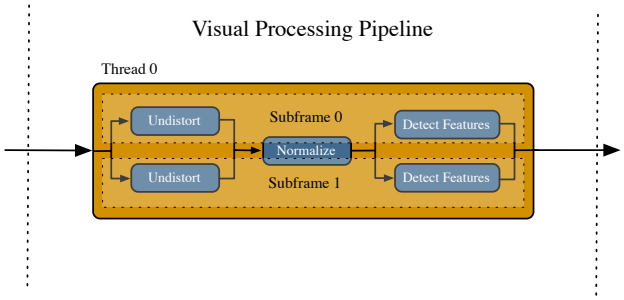
All threads in the visual processing pipeline always contain the same sequential code. To perform the various vision-based tasks, different pipelines are used. They can be stopped and activated according the robots current task.

Stereo frames are de-queued from the inbound queue as soon as visual processing begins. Results from the outbound queue are integrated together with various data to control the mobile robot platform.

The vision algorithms introduced in the following chapters employ this pipeline approach for real-time vision-based navigation and environment mapping.



(a)



(b)

Figure 3.5: *(a) Visual processing pipeline. (b) Visual processing example.*

CHAPTER 4

FEATURE ACQUISITION AND INTEGRATION

4.1 Problem Description

Our vision-based navigation and environment mapping approach exploits natural visual features as landmarks. These landmarks need to be detected and described by a feature vector to enable tracking and unique view-independent matching. The main computational effort is spent on feature tracking and description. Putative image feature matches can be used subsequently to determine the 3D coordinates of landmarks and thus to estimate potential camera view transformations.

Our mobile robot platform has a non-stationary stereo camera that can be rotated independently. This enables the acquisition of features over a wide angular range at a single robot position but makes it necessary to integrate landmarks from many different camera angles. In most cases the true rotation angle of the camera cannot easily be determined mechanically due to the lack of an appropriate feedback system. Hence it is necessary to be able to approximate the camera rotation directly based on the detected landmarks. However, estimating the camera rotation in a robust way increases the computational complexity significantly. Furthermore, real-world indoor environments have proven to be extremely challenging because of the lack of dense textured features. Poor feature extraction and sparse landmarks result easily in a lack of suitable matches and wrong feature integration.

We address the problem of real-time acquisition and integration of visual features in real indoor environments. Our optimized feature detection and matching methods are suitable for online processing on COTS[1] vision-based robot platforms with limited computational resources. Despite focusing mainly on system performance, we achieve excellent accuracy in feature integration even in the presence of outliers or sparse landmarks. Our contributions include:

- Adaptive feature detection based on the *FAST* corner detector
- Improved *SIFT* based feature description for real-time processing
- Fast feature matching exploiting spatial and temporal coherence
- Stable and accurate feature integration from a non-stationary stereo camera
- Experimental comparisons to standard methods used in the area

4.2 Related Work

The acquisition of landmarks in images is generally carried out in two steps:

1. Detection of suitable visual features that can be used as landmarks.
2. Description of the features with a feature vector that uses local image neighborhood information.

A number of methods for both steps have been proposed in the past. We focus here on the ones primarily used in natural feature acquisition.

The *Harris* [Harris and Stephens, 1988] and *SUSAN* [Smith and Brady, 1997] corner detectors deliver high quality features due to their strong invariance to rotation and translation. A simple image intensity patch is usually used for the feature description stage [Jonathan and Zhang, 2007], but such a descriptor is not invariant, restricting the possible view transformations. Note that both corner detectors have been reported to be computationally expensive [Rosten and Drummond, 2006]. They have been combined with more elaborated feature descriptors [Ballesta et al., 2007] at a reduced overall performance.

KLT developed by Kanade, Lucas and Tomasi [Shi and Tomasi, 1994] extracts image features that are adequate for tracking. *Normalized cross correlation* (NCC) tracks these features in subsequent images. Tracking is successful if an affine transformation between the current and the original image patches can be found. Lost features are replaced by new features to keep a constant feature count over

[1] Commercial off-the-shelf

multiple images. Tracking has the advantage that features can still be followed even after the feature detector ceases to detect them. Thus, feature localization is very accurate, but if the search area is not sufficiently limited NCC exposes a poor performance with increasing image size (q.v. Section 4.10).

The most popular feature detection and description approach is *SIFT* (Scale-Invariant-Feature-Transform) [Lowe, 2004]. Visual features are detected as local extrema of the *Difference of Gaussian* (DoG) over scale space. The SIFT descriptor is rotation and scale invariant. Landmarks can be identified even after substantial view transformations. The SIFT descriptor uses a N^2 descriptor array computed from a subregion around the feature with M - gradient orientations. The most common representation with $N = 4$ and $M = 8$ results in a 128 element feature vector. Computationally, SIFT is one of the most expensive descriptors though achieving an excellent invariant feature description.

More recently *SURF* (Speeded up Robust Features) [Bay et al., 2006] has been proposed. In SURF, feature detection is based on the *Hessian matrix* while the descriptor uses sums of 2D Haar wavelet responses in a 4×4 region resulting in a 64-dimensional feature vector. SURF is also scale and rotation invariant exposing a similar description quality as SIFT at a better performance [Bauer et al., 2007].

For registration, the acquired landmarks are matched by calculating the Euclidian of their descriptors. The nearest match must be closer than a certain percentage to the second nearest match, as matches caused by noise will have multiple noisy matches. To avoid a brute-force comparison of the descriptors, k-d trees, spatial hashing, and epipolar matching can be employed.

Once putative matches are found, 3D coordinates of the landmarks can be calculated by triangulation from different images. Features are spatially integrated estimating the view transformation from corresponding 2D image points or corresponding 3D world points.

Based on the correspondence of 2D image points the fundamental matrix F can be determined [Hartley and Zisserman, 2004]. Several algorithms have been proposed for this purpose [Zhang and Kanade, 1998]. With the fundamental matrix we can estimate the essential matrix E using the camera calibration matrix K:

$$E = K^T F K \tag{4.1}$$

Decomposing the essential matrix using singular value decomposition (SVD)

$$E = USV^T \tag{4.2}$$

the rotation matrix

$$R = UWV^T \tag{4.3}$$

where

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

can be extracted and the rotation angles recovered. The translation is given by

$$T = V \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{4.5}$$

Several RANSAC-schemes [Fischler and Bolles, 1981] exist to eliminate outliers, as they considerably influence the estimated fundamental matrix.

The *iterative closest point* (ICP) algorithm [Besl and McKay, 1992] computes the view transformation from corresponding 3D world points. Arun et al. [Arun et al., 1987] use a SVD of the covariance matrix of 3D point data. As this algorithm gives sometimes a reflection instead of rotation, Umeyama [Umeyama, 1991] proposed an approach that always produces the rotation.

Estimating the view transformation based on 3D point correspondences suffers from a major drawback. Triangulations are much more uncertain in the depth direction resulting in poor transformation estimation. Therefore, the estimation based on 2D image points could give a more precise view transformation.

Our system for the online acquisition and integration of image features avoids expensive computational feature detection and tracking by using the *FAST* corner detector proposed by Rosten et al. [Rosten and Drummond, 2006] combined with a modified reduced SIFT descriptor [Lowe, 2004]. Registration and matching of features in real-time is achieved by exploiting optimized spatial hashing [Teschner et al., 2003]. Features are spatially integrated by estimating the view transformation on corresponding 2D image points directly, using a rather simple but stable algorithm. Our system provides very accurate results at comparatively low computational cost that no other previously proposed method is capable of in our indoor environments.

4.3 System Setup and Visual Processing

The Bumblebee 2 stereo camera is attached to a pan/tilt system permitting an absolute rotation of $130°$ (q.v. Section 2.2.2). Mechanically, the rotation angle can be controlled from $-65°$ to $+65°$. However, no exact feedback or control is possible for accurate setting of intermediate angles. The center position and orientation of the camera is defined to be parallel to the ground plane and perpendicular to the robots center line as seen in Figure 4.1.

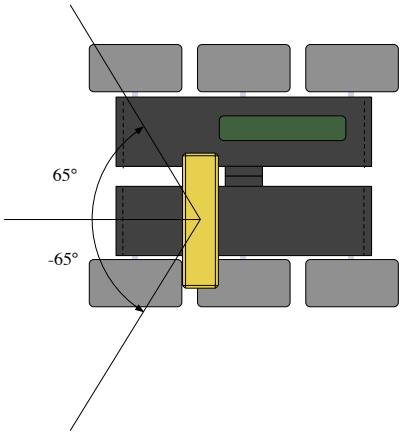


Figure 4.1: *Robot platform FOV.*

Feature points are acquired from the rotating stereo camera and spatially integrated. This includes detection, description and matching of feature points as well as accurate estimation of rotation angles. The setup of the visual processing pipeline is outlined in Figure 4.2 and will be explained in detail below.

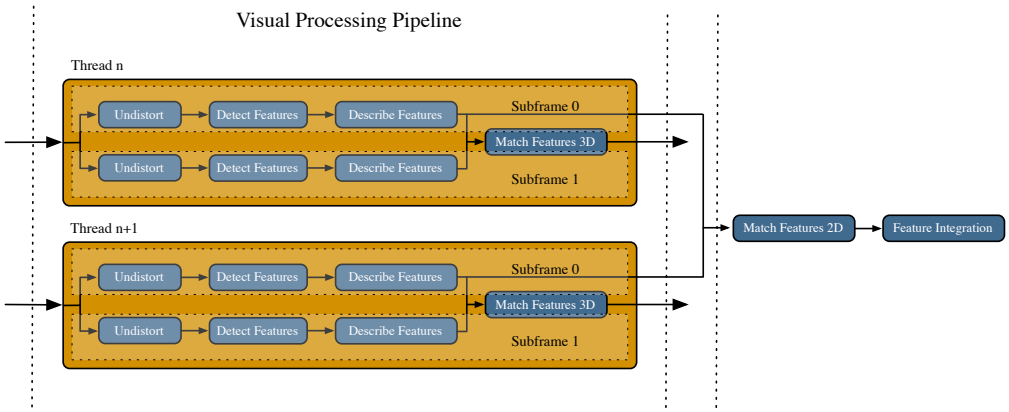


Figure 4.2: *Visual processing pipeline setup for real-time feature acquisition and integration.*

4.4 Distortion Correction

Acquired images are basically affected by barrel or pincushion distortions. These distortions are caused through wide angle and zoom lenses respectively. With

increasing distance from the image center, distortions are more pronounced.

We correct camera lens distortions with a reverse lookup. The lookup-table is pre-generated from camera calibration data and maps pixels from undistorted to distorted image positions. To obtain the undistorted pixel value, neighboring pixels at the distorted position are bi-linearly interpolated (Figure 4.3).

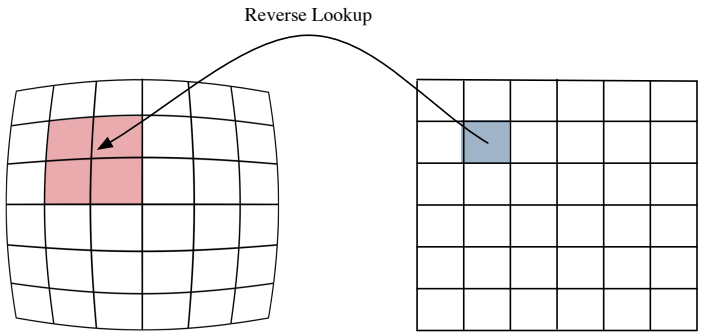


Figure 4.3: *Barrel distortion correction with bi-linear interpolation.*

4.5 Feature Detection

Our feature detection is based on the *FAST* corner detector with non-maximum suppression for feature detection [Rosten and Drummond, 2006]. FAST is a high quality corner detector that significantly outperforms other existing algorithms. The principle of FAST is to examine a small patch around a candidate image point to see if it looks like a corner. This approach is efficiently implemented by a segment test algorithm improved through machine learning. We use the 9-point variant of this corner detector for our feature detection stage, as it provides optimal performance.

Regardless of being optimized for performance, the FAST corner detector is invariant to substantial view transformations and independent of the feature type. Its major disadvantage lies in the dependence on a user defined threshold. Nevertheless, this feature detector exposes such a great performance so that it can be used for adaptive feature detection. Instead of defining an image-based threshold we can define a desired feature count. The optimal threshold can then be found in a few iterations using the algorithm shown in Figure 4.4.

First we pre-define threshold step sizes that proved to be appropriate for fast threshold determination (1). We then iterate until a user defined limit (2), run the feature detector (3), and return the features if they lie within a 10% threshold of the desired feature count (4, 5). If the feature number differs to a greater extent the threshold is adjusted by a step value in the direction of the desired feature count

(7, 8). Should we pass the target features we start reducing the step size and repeat steps (2-9). If no appropriate threshold can be found after exceeding the iteration limit, the feature search with the closest count is returned. This way we aim for a constant feature count by adaptive thresholding.

Section 4.10.1 shows that adaptive feature detection has only a marginal impact on the overall performance, whereas keeping a constant feature count is an important component for feature description and integration. Too sparse features result in an uncertainty in the estimated view transformation, while an excessive number of features increase significantly the time spent on feature description and thus breaking the real-time constraint of the robot platform.

```
1  threshold step = {10, 5, 2, 1};
2  while iteration limit not reached
3      run feature detector;
4      if feature count within 10% of target features
5        return features;
6      (* adjust feature detection threshold *);
7        determine threshold step sign;
8        threshold ± = threshold step;
9        if passed target features
10         begin reducing threshold step;
11 endwhile
```

Figure 4.4: *Adaptive feature detection procedure.*

4.6 Feature Tracking

Detected features could be tracked between image frames by cross correlation or sum-of-squared differences (SSD) matching. We avoid actual feature tracking because the involved search in high resolution images would be much more time-consuming than our fast feature detection, description and matching. Tracking is eventually achieved by matching features in multiple images as outlined below in Section 4.8.

4.7 Feature Description

Feature detection is directly followed by feature description. Feature description is a fundamental part as it is used to associate landmarks from different views.

Wrong associations between landmarks will result in inaccurate feature registration and integration, thus each detected feature needs to be assigned a unique invariant descriptor.

As noted in Section 4.2, SIFT achieves an excellent invariant feature description at the expense of decreased performance. Though quite fast implementations exist that employ SSE and OpenMP (Open Multi-Processing) claiming speed improvements of a factor of 6 over Lowe's standard approach [OpenSource, 2008], they are still not sufficient for real-time usage on high-resolution images.

Referring to SIFT, it is always considered in its most expensive variant with a 128-element feature vector. A smaller variant of the descriptor exists that performs only 8% worse in terms of feature association than the full version [Lowe, 2004]. This variant uses a 3×3 descriptor array with only 4 gradient orientations. The resulting 36-element feature vector is much faster to compute and suits the real-time constraints of our system.

In detail, we choose a subregion around the feature that is 15×15 pixels wide. This subregion is divided into 5×5 pixel wide regions. Keeping close to the original implementation we calculate the gradient orientation and magnitude for each pixel in the subregion. Weighted by the distance from the region center and the magnitude, gradient orientations are accumulated into 4 gradient orientation histograms (Figure 4.5). The descriptor is normalized to unit length to reduce effects of illumination changes. To reduce the influence of large gradient magnitudes, descriptor values are clamped to 0.2 and re-normalized as proposed in [Lowe, 2004].

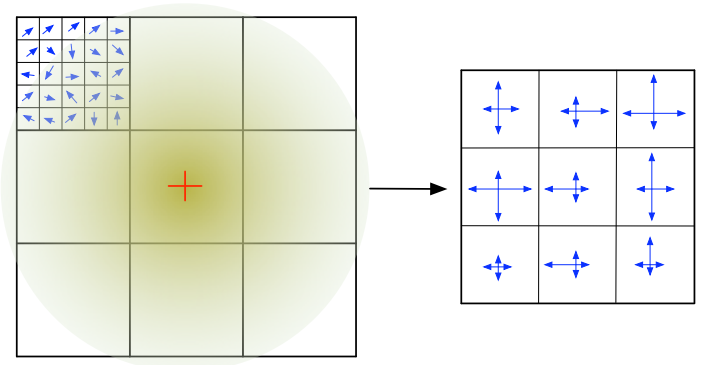


Figure 4.5: *36-element reduced SIFT descriptor creation.*

We implemented the descriptor calculation without any specific multi-processing or SSE extensions. Nevertheless, we achieve real-time performance on high resolution images (see Section 4.10.1). Figure 4.6 shows the outline of our *real-time SIFT* (RTSIFT) method.

During the initialization of RTSIFT we pre-calculate square root and arc tangent lookup tables (2, 3). Before considering individual descriptors we calculate the x- and y-image gradients once (6, 7). This can be performed more efficiently on the entire image than on separate sub-images as feature regions tend to overlap. To describe an image feature we lookup the gradient magnitudes and orientations for each pixel in the feature's subregion (9, 10) and accumulate them after Gaussian weighting into the descriptor array (11). A list of descriptors is eventually returned (13).

```
1  (* Initialization *)
2    pre-calculate square-root lookup table [256, 256];
3    pre-calculate arc-tangent lookup table [512, 512];
4
5  (* Descriptor calculation *)
6    calculate image x - gradients;
7    calculate image y - gradients;
8    for each detected feature do
9      lookup gradient magnitudes in square-root table;
10     lookup gradient orientations in arc-tangent table;
11     accumulate weighted orientations to descriptor array;
12   endfor
13   return descriptors;
```

Figure 4.6: *Feature description process.*

4.8 Feature Matching and Outlier Removal

Feature matching associate's landmarks from different views that correspond to the same feature. The similarity of two features is defined by the Euclidian distance of their feature descriptors. Comparing feature descriptors using a brute-force approach leads to a matching time that is linearly dependent on the number of features. Using spatial and temporal coherence we can avoid the linear dependency and considerably decrease the time spent on feature matching and reduce outliers at the same time.

To limit the number of descriptor comparisons we use *spatial hashing* [Teschner et al., 2003]. Spatial hashing divides the image space into grid cells (Figure 4.7). Each of these grid cells is assigned with a unique hash value. The grid cell size influences the number of landmarks that will reside in a cell. For matching, only landmarks within the same and neighboring grid cells are considered. The optimal grid cell size depends on the matching type. In stereo matching the grid size is

set according to the expected depth range and thus the corresponding disparities. To estimate view transformations we use knowledge about the pixel displacement given by a certain robot or camera movement. We set the grid cell size to the expected maximum disparity or pixel displacement.

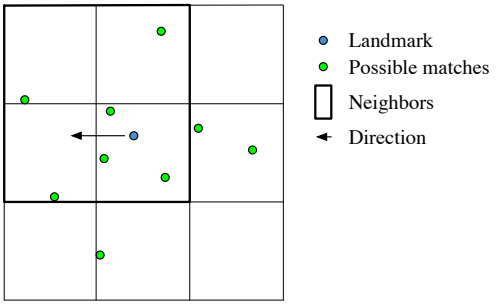


Figure 4.7: *Spatial hashing to limit feature search.*

Our matching method is outlined in Figure 4.8. After setting the grid size according to the matching type the hash table is initialized with landmarks. We use a hash function (4) with the prime numbers $p1 = 73856093$, $p2 = 19349663$ and the hash table size n set to the landmark count. For each landmark (3) its grid position and the corresponding hash value are generated (4). The landmark's Euclidian distance is calculated to all landmarks corresponding to the same hash value (5). If a match is not found (6), neighboring grid cells are searched (7). These neighbor grid cells depend on the landmark position and the direction of the pixel displacement. Features are only associated if they are closer than 50% of the second closest match (9).

Spatial hashing contributes greatly to the reduction of outlier matches as the spatial coherence constraint generates fewer mismatches.

Feature matching can be improved further by considering temporal coherence. Robot and camera motion are nearly constant over a short time, thus it is possible to predict a landmark position based on previous matches. This is done by *epipolar matching*. As mentioned in Section 4.2 the fundamental matrix F can be determined based on 2D image point correspondences. Having F, we can estimate the epipolar lines along which landmarks should move. Matching is hence reduced to features that are near an epipolar line e' as illustrated in Figure 4.9. But uncertainty in the estimation of the fundamental matrix leads to wrong predictions.

As the camera movement of our robotic platform at a single position is limited to rotations, landmarks move either horizontal or vertical in image space, and we can reduce the problem to a simple 1D lookup table as shown in Figure 4.10. Mismatches between similar features that remain after feature matching are handled

```
1  set grid size according matching type;
2  initialize hash table;
3  for each described feature do
4    generate hash = (x · p1 ⊕ y · p2)  mod n;
5    compare feature descriptors inside grid cell;
6    if feature not found
7      search neighboring grid cells;
8    if match 50% closer than second match
9      associate features;
10 endfor
```

Figure 4.8: *Feature matching algorithm.*

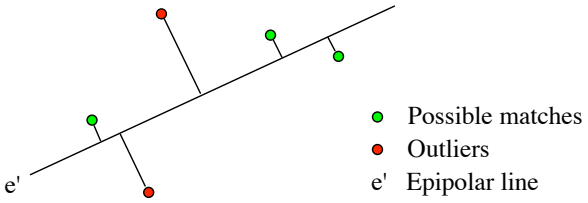


Figure 4.9: *Epipolar matching.*

during the following feature integration stage.

4.9 Feature Integration

Assuming a calibrated camera with known intrinsic parameter matrix (Equation 4.6), f being the focal length and X_c, Y_c the camera image center, we can easily triangulate the 3D position of associated landmarks (Equations 4.7-4.8).

$$C = \begin{pmatrix} f & 0 & X_c \\ 0 & f & Y_c \\ 0 & 0 & 1 \end{pmatrix} \tag{4.6}$$

The parameter b represents the camera baseline and d the stereo disparity (Equation 4.7).

$$z = \frac{b \cdot f}{d} \tag{4.7}$$

$$x = \frac{(x_i - X_c) \cdot z}{f}, \quad y = \frac{(y_i - Y_c) \cdot z}{f} \tag{4.8}$$

```
1  (* Initialization *)
2    create lookup table [image y size];
3    for all matched landmarks do
4      lookup table [landmark y position] = 1;
5
6  (* Consecutive matches *)
7    if lookup table [feature y position and ± 1] = 0
8      skip feature;
```

Figure 4.10: *1D lookup table for horizontal landmark movement.*

As noted in Section 4.2, estimating the view transformation from 3D points is unreliable due to depth uncertainties. Hence we estimate the camera movement based on 2D points from monocular images at continuous time steps.

Standard algorithms for estimating the fundamental matrix F [Zhang and Kanade, 1998] showed to be inappropriate for feature integration on our robotic system, due to sparse feature correspondences.

The angle and axis of rotation recovered from the fundamental matrix varied strongly with the standard algorithms and in most cases misleadingly a translation was found instead of a proper rotation. Furthermore, the fundamental matrix is very sensitive to slight changes in the landmarks' 2D positions and to outliers even when using RANSAC.

The camera on our robotic platform is rotating in angular steps $\leq 1^\circ$. Therefore, we need a more robust feature integration approach that is able to reliably estimate even small angles. For the sake of simplicity we consider in the following a camera rotation around the y-axis, as an x-axis rotation can easily be derived from the given equations.

For each pair of associated landmarks we calculate the rotation angle ϕ using the cameras intrinsic parameters and trigonometric relations as indicated in Figure 4.11.

The rotation angle ϕ is found based on the projected image positions (p_i, p_i') of the landmarks (P_i, P_i'). According to the projection of p_i and p_i' relative to the camera's image center X_c we determine the intermediate angles α and β as:

$$\alpha = atan(\frac{p_i' - X_c}{f}), \quad \beta = atan(\frac{X_c - p_i}{f}). \tag{4.9}$$

The angle ϕ is then easily found from

$$\phi = \alpha + \beta. \tag{4.10}$$

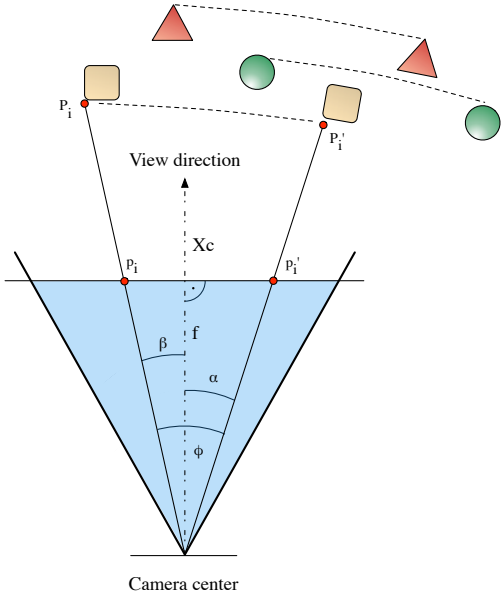


Figure 4.11: *Relation of camera rotation angles to disparities.*

Given a set of individual rotation angles ϕ we need to find a common angle that agrees with most landmarks and takes outliers into account which were not eliminated during the previous feature matching stage.

Our solution to this is the following: First, we find the angle ϕ that corresponds to the majority of angles inside the set within a 10% threshold. This is done by simply testing each angle ϕ against all others. Second, landmarks that are not consistent with ϕ are considered to be outliers and excluded from estimating the rotation angle. The angle ϕ is finally used as starting point for iterative, non-linear least squares Levenberg-Marquardt optimization under the condition that the error $E(\phi)$ becomes minimal.

$$E(\phi) = \sum_{n=0}^{N} [p'_i - f(p_i, \phi)]^2$$

The angle obtained through this optimization is used for 3D feature integration according to triangulation and the estimation of the cumulative rotation angle (see Section 4.10.2).

Even though we are assuming that the camera is rotating around its center it is in fact shifted by $30mm$ off the real rotation axis on our robotic system. Figure 4.12 shows that the error introduced through this approximation can be

neglected. For a depth range of $1 - 5m$ and camera offsets $0 - 30mm$ the pixel projection error is in the sub-pixel range < 0.5 pixel.

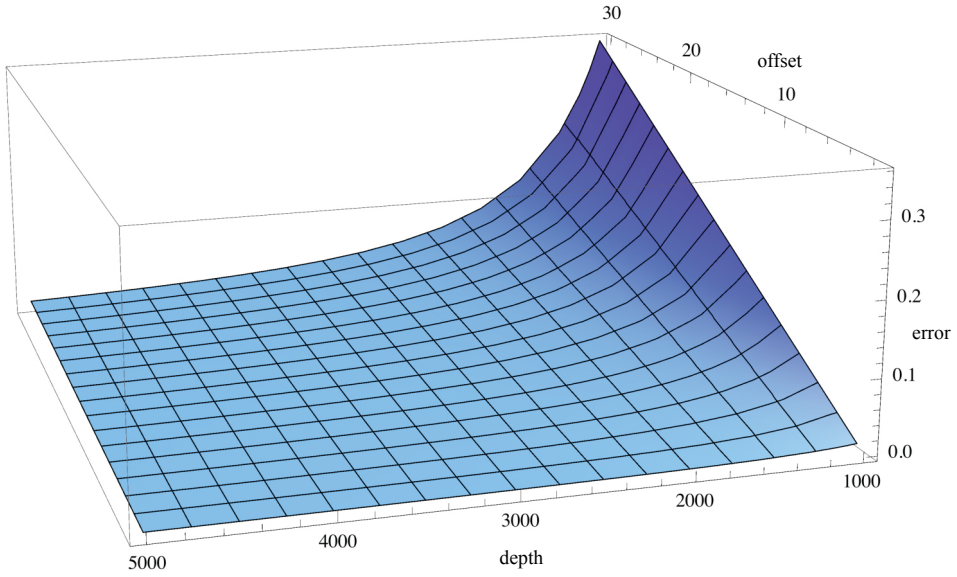


Figure 4.12: *Pixel error introduced for varying offsets from the rotation center and depth ranges.*

4.10 Experimental Results

We tested our feature acquisition and integration on the mobile robot platform that uses an Intel Core Duo 2, 2GHz CPU for vision processing. The results presented below are therefore using four pipelines unless stated otherwise.

4.10.1 Performance

To compare the performance of our RTSIFT implementation with adaptive feature detection to different standard methods, a real-world indoor sequence of 100 frames was recorded. On this pre-recorded sequence we compared RTSIFT [Hübner and Pajarola, 2009] to KLT [Birchfield, 2007], Fast SIFT [OpenSource, 2008], and SURF [Bay, 2006]. Timings are given in Figure 4.13 for the feature detection and description and for three different image resolutions. RTSIFT clearly outperforms any standard method for the feature detection and description on high-resolution images. We achieve 18*fps* at a resolution of 1024×768, while SURF (2.6*fps*), KLT (0.8*fps*) and Fast SIFT (0.5*fps*) are significantly slower.

We additionally evaluated the influence of the adaptive feature detection (AFD) on the performance of the feature detection, as well as on the number of detected

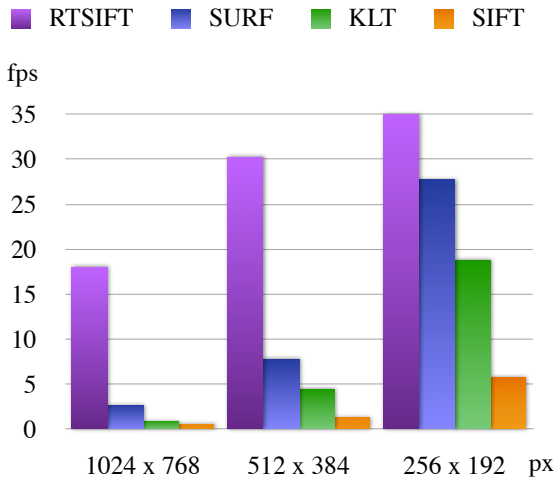


Figure 4.13: *Performance of feature detection, description and angle estimation for different algorithms.*

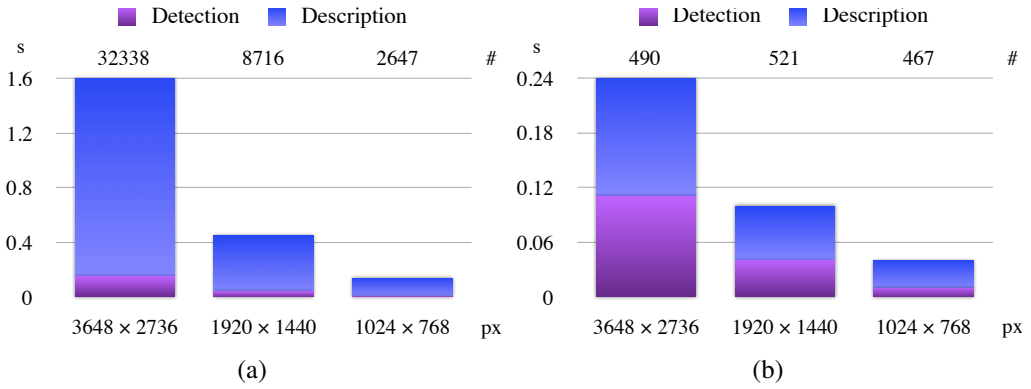


Figure 4.14: *Feature detection and description (single-threaded). (a) Original FAST feature detector. (b) Adaptive feature detection.*

features and the resulting description time. For the adaptive detection the number of target features was set to 500 and the iteration limit to 6. The initial threshold for FAST feature detection was set to 15. Figure 4.14(a) shows the original FAST corner detector compared to our implementation with adaptive feature detection (Figure 4.14(b)). With increasing image size the number of detected features increases proportionally when using the original FAST corner detector. While the excessive number of features is not necessarily beneficial, this reduces the perfor-

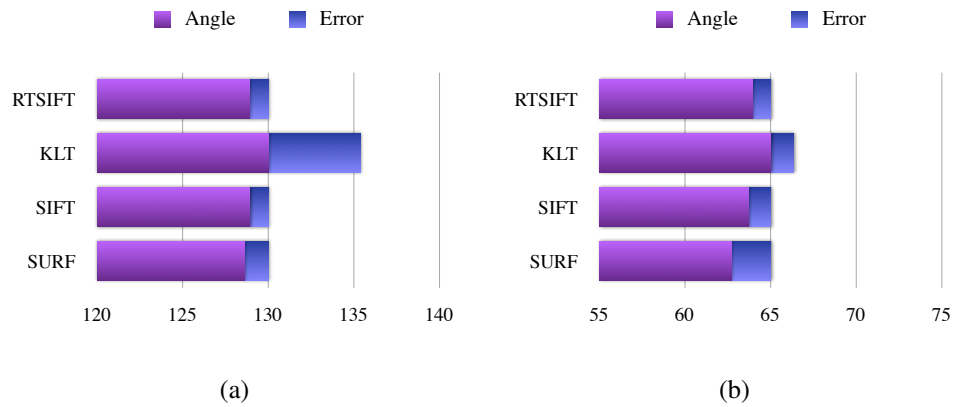


Figure 4.15: *Accuracy of estimating the cumulative rotation angle. (a)* 130° *camera rotation. (b)* 65° *camera rotation.*

mance of the subsequent feature description stage. Our adaptive feature detection keeps the number of features near a given constant and thus guarantees fast computation for feature description.

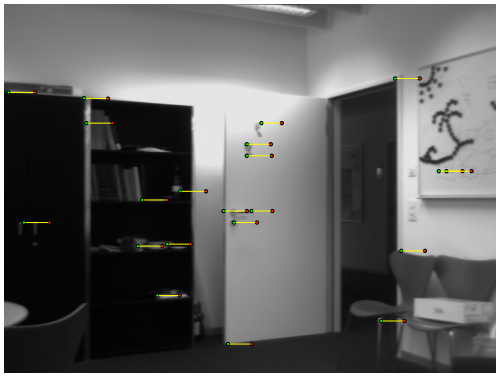
We noticed that pre-calculating x- and y-gradients during the feature description process (Figure 4.6) for high-resolution images ($> 1024 \times 768$) is in fact more time consuming than on-the-fly calculation. Consequently we are using the most appropriate approach.

4.10.2 Accuracy

The accuracy of the system is an important factor for the feature integration. We tested our method with different pre-defined rotation angles. While rotating the camera, landmarks are continuously acquired and intermediate rotation angles are estimated. The resulting accumulated angle should ideally correspond to the real camera rotation. Figure 4.15(a) and 4.15(b) show the error in estimating the cumulative rotation angle for the different methods. While all methods achieve a rather low error for estimating the rotation angle, using our feature integration, RTSIFT is by a factor of ≈ 7 significantly faster.

In Figure 4.16 we show an example of the real-world environment with matched features in the left-right stereo images as well as matched features in subsequent frames over time. Our novel method for feature acquisition and integration is able to extract sufficient landmarks from the real-world indoor environment, to establish correspondences and to integrate the landmarks with high accuracy.

(a) (b)



(c)

Figure 4.16: *(a) Left-view image with detected landmarks in green. (b) Right-view with matched stereo correspondences. (c) Right image at $t+1$ with matched correspondences over time.*

CHAPTER 5

NAVIGATION

The adapted behavior of robots in an unknown environment still represents a challenging research task. Robots are most commonly tested in man-made or specifically prepared environments, whereby external impact parameters are known and the robot's behavior is predefined. Alteration of the environment can strongly affect the robot and cause an undefined behavior. Hence environment-adapted behavior based on a regulatory feedback system, e.g. vision, similar to the human's visual and cognitive system is desirable, especially for robot navigation. Visual feedback and a defined navigation objective should control the robot's hardware behavior.

The first basic behavior needed for a semi-autonomous robot is obstacle detection and avoidance.

5.1 Obstacle Detection and Avoidance

Various sensors and systems exist to determine the robots distance to surrounding objects. They can be divided into two major groups: Hardware sensors that directly provide range data and vision-based systems that employ a single- or multiple camera configuration together with computer vision methods to estimate the range. Hardware sensors that work with infrared light, lasers or ultrasound to obtain range information are extensively used for mobile robots and in particular real-time navigation. Vision-based obstacle detection, like stereo-vision, optical flow, depth from focus and active scene illumination is typically limited to robot

platforms with sufficient high computational power to process and analyze the camera images. Unfortunately none of these sensors or systems is perfect.

5.1.1 Problem Description

Infrared and sonar sensors are lightweight, cheap and have low power consumption. They allow simple system implementation and fast obstacle detection at a frequency of several hertz per second. Accuracy and reliability though are limited.

The detection range of infrared sensors is restricted ($\leq 1.5m$). Within this short range the position of an obstacle can be determined quite exactly. Ambient light, especially sunlight, specular surfaces and cross-firing cause interferences.

Sonar sensors can detect obstacles inside a wider range ($\approx 10m$), at the cost of a poor angular resolution. This makes it difficult to find the obstacle's exact position. Furthermore the obstacle orientation and shape can influence the sonar and lead to erroneous measurements.

Laser rangefinders provide a superior obstacle detection in real-time at a much higher price. Size, weight and power consumption of these sensors are outside the constraints of our mobile robot platform.

Cameras used for vision-based obstacle detection provide more comprehensive information about the environment. Most computer vision methods that estimate range information based on camera images require a well textured and lit environment to perform properly. Moreover they are computationally expensive and therefore generally not applicable to mobile robots.

5.1.2 Related Work

The obstacle avoidance problem in robotics has been researched extensively and there are well established algorithms. Most of these algorithms are developed for large robots with expensive, specialized sensors and powerful computing platforms [Darms et al., 2009].

Vision based obstacle detection and avoidance is becoming a popular alternative to sonar- and infrared sensors, as it provides a better resolution at a reasonable price. Image processing though, is still a very computationally intensive task. Fast methods that have been proposed for small, low-cost mobile robot platforms range from ground segmentation [Viet and Marshall, 2007], [Ulrich and Nourbakhsh, 2000], the usage of structured light [Ilstrup and Hugh Elkaim, 2008], [Wei et al., 2009] to laser projection [S. Soumare, 2002]. Most methods work with a reduced image resolution, negating the major advantage of a vision-based system.

We developed a system that is feasible for autonomous mobile robots considering their limited resources, weight and power limitations that is built from low

cost off-the-shelf components. For this system we combined sonar based obstacle detection with a vision-based refinement supported by active-illumination.

5.1.3 System Setup and Visual Processing

Our mobile robot is equipped with a camera and two sonar sensors. The sonar sensors are facing in opposite directions, one together with the camera in front and on the back. Both sonar sensors are active at the same time, nevertheless erroneous readings caused by cross-firing are rare.

The only information that can be obtained by the sonar sensors is the distance to the closest point of the obstacle that reflected the wave, back to the sensor. This is not sufficient to characterize the obstacle, furthermore the poor angular resolution makes it impossible to determine to which direction the measurement corresponds.

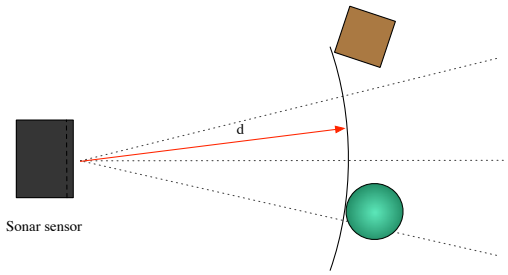


Figure 5.1: *Uncertainty in sonar based obstacle detection. Obstacles at different positions give the same distance reading d.*

Figure 5.1 illustrates a situation where two possible obstacles produce the same range reading. A sonar range value hence only defines a region in which every point is a possible location for the detected obstacle. This region takes the form of one arc centered in the sensor, with the radius equal to the range reading. To overcome this disadvantage, we are using a camera in combination with a pulsed $650nm$ line-laser as soon as an obstacle is detected by the sonar.

The laser projects a plane parallel to the ground in front of the robot platform. The camera is automatically tilted to observe the plane projection. Based on this projection, the exact position and shape of the obstacle can be determined. Our system setup is shown in Figure 5.2.

For finding the laser in the image an optical band-pass filter is not an option, as it would limit the camera's usage solely to obstacle detection. Moreover this filter might limit the passing light to a narrow band around $650nm$, but most objects in an natural environment reflect daylight at this wavelength. This makes it difficult to clearly distinguish the laser in the image.

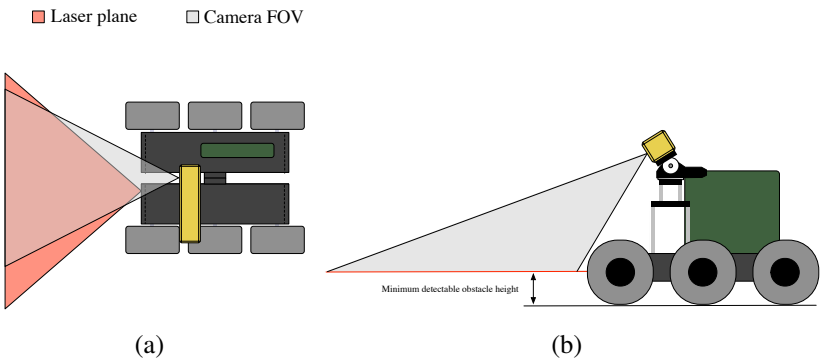


Figure 5.2: *Robot system setup for vision-based obstacle detection.*

We use image differencing for finding the laser projection. An image is taken before the laser is activated and subtracted from the image with the activated laser. The resulting difference image is quite close to the laser projection. Camera noise still has to be excluded by a threshold filter. Figure 5.3 shows an example of two consecutively acquired images and the resulting difference image.

The projection of the laser line is clearly visible. We should mention that the image differencing was applied to the red channel of the images, where the highest response signal can be expected.

With a pre-calibration of the system, distance values can be directly extracted from the image. Calculation of the distance values is based on optical triangulation according to Figure 5.4.

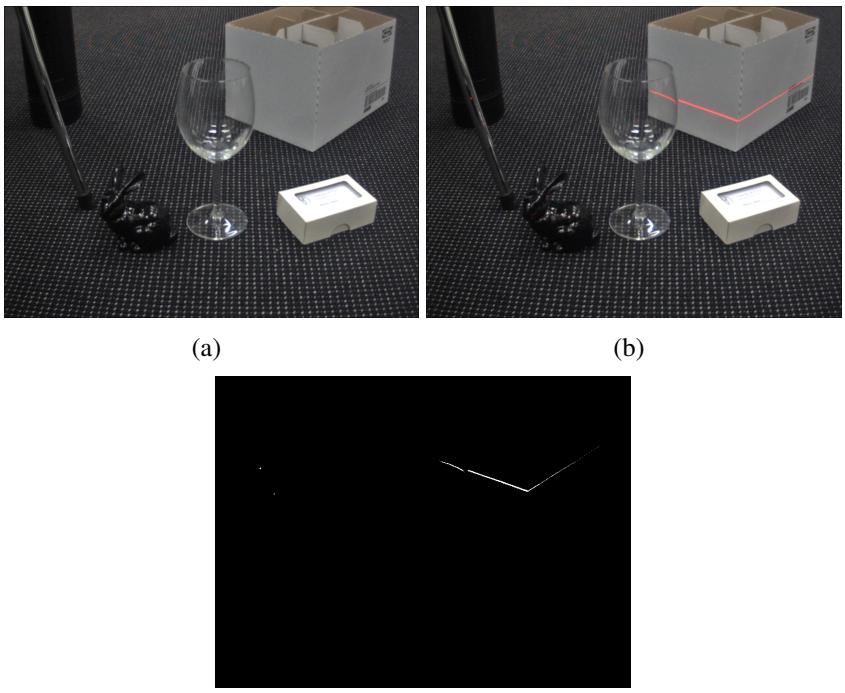
At first we have to determine the angles α_z and α_0 (Equation 5.1 and 5.2). D_z corresponds to the distance of the camera center to an object projecting to the image center, while D_0 is the distance of the camera center to an object that projects to the first image row ($k = 0$). The distance between the camera and the line-laser is defined as b. Based on α_z and α_0 we can calculate the half sensor size d (Equation 5.3). The focal length of the camera defines f.

$$\alpha_z = arctan(\frac{D_z}{b}). \tag{5.1}$$

$$\alpha_0 = arctan(\frac{D_0}{b}). \tag{5.2}$$

$$d = f \cdot tan(\alpha_z - \alpha_0). \tag{5.3}$$

Our image has M columns and N rows (768×1024). The increment between two columns is defined by



(a) (b)

(c)

Figure 5.3: *Robot system setup for vision-based obstacle detection. a) Image scene. b) Image scene with activated laser. c) Difference image.*

$$d_k = \frac{2kd}{M}. \tag{5.4}$$

The angle α_k between the projected laser plane and the reference plane is defined depending on the position to the camera's optical axis ($0 \leq k \leq M/2$ or $M/2 < k \leq M-1$) by

$$\alpha_k = \alpha_z - \phi'_k \quad \text{or} \quad \alpha_k = \alpha_z - \phi''_k. \tag{5.5}$$

with

$$tan(\phi'_k) = \frac{d - d_k}{f} \quad \text{or} \quad tan(\phi''_k) = \frac{d_k - d}{f}. \tag{5.6}$$

placing 5.4 in 5.6

$$tan(\phi'_k) = \frac{d * (M - 2 * k)}{M * f} \quad \text{or} \quad tan(\phi''_k) = \frac{d * (2 * k - M)}{M * f} \tag{5.7}$$

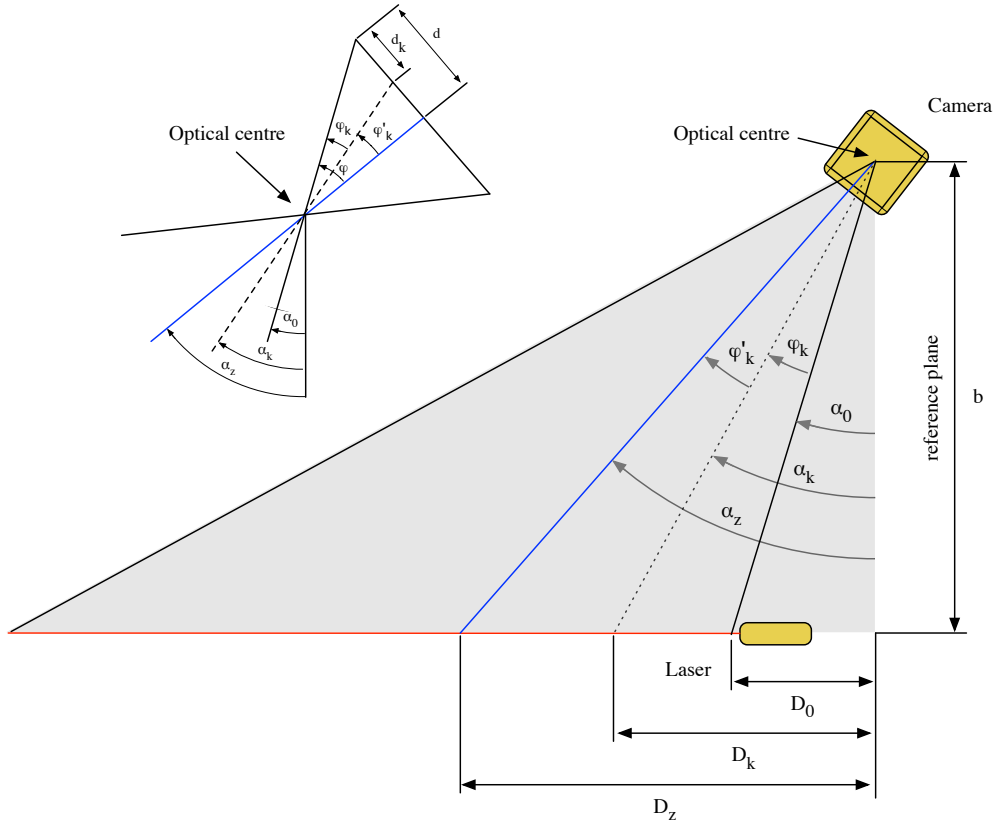


Figure 5.4: *Determining the distance and shape of obstacles with optical triangulation.*

The distance D_k can be determined with the angle α_k.

$$D_k = b * tan(\alpha_k). \tag{5.8}$$

As the values of the function D_k have a nearly exponential distribution, we can approximate distances by fitting an exponential function (Equation 5.9, a, b and c are the parameters) through the constant values D_0, D_z, D_{M-1} with image rows k normalized (Figure 5.5). The distance D_k is then simply a value of this function.

$$D_k = a \cdot e^{bk} + c. \tag{5.9}$$

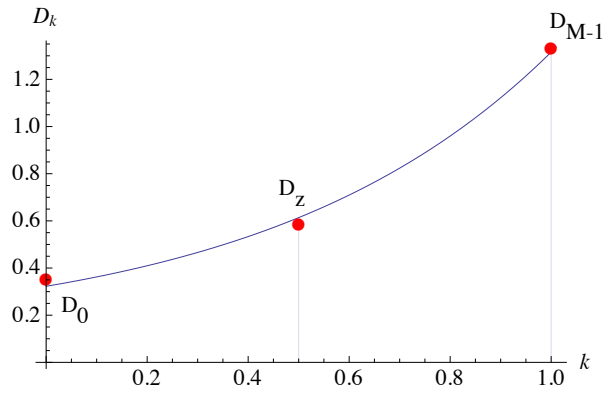


Figure 5.5: *Exponential fitting image rows k to distance values D_k.*

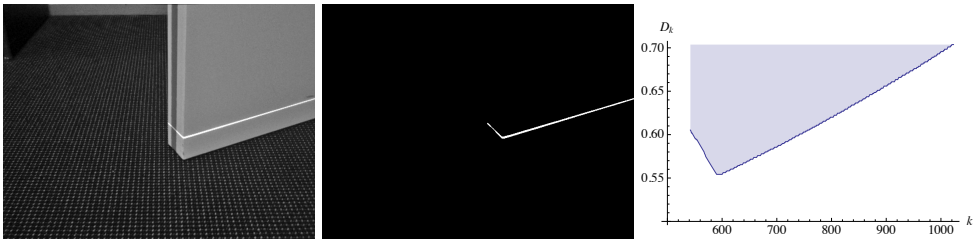
5.1.4 Experimental Results

We placed the robot in an unmodified indoor office environment to validate our obstacle detection. The sonar sensor activated the camera if an obstacle was detected to be closer than $60cm$. At this distance the robot stopped, tilted the camera and acquired two images, one with and one without the laser. The laser projection was extracted from the resulting difference image.

Our camera field of view covered a depth range of $0.35m - 1.33m$, with a resolution of 768 pixels. Depth values are non-linearly distributed and have a range between $0.4mm$ and $3mm$ per pixel. However the real positioning error for the obstacle detection is significantly higher, as it is very difficult to exactly measure the distance constants D, and depth values are only approximated by the exponential function.

Simple and effective obstacle avoidance was implemented, that rotates the robot always in the direction of free navigable space. We could observe a reliable avoidance behavior and a long term collision free navigation. Figures 5.6, 5.7 and 5.8 show examples of detected obstacles (a), the extracted laser projection (b) and their approximated shape and distance (c).

During our experiments we observed several artifacts in the detection of obstacles. Small objects below the laser plane (e.g. business card box in figure 5.3) are not detected. Though in most cases, the robot is able to pass over them. Reflections are a major contributing source to erroneous readings in the system. The laser projection is only partially visible on reflective surfaces, making them difficult to detect (e.g. metallic chair leg in figure 5.3). Transparent objects can distort the laser projection and give readings not associated with the shape of the real obstacle (e.g. glass in figure 5.3). Dark surfaces may not reflect enough energy



(a) (b) (c)

Figure 5.6: *Door.*

(a) (b) (c)

Figure 5.7: *Column.*

(a) (b) (c)

Figure 5.8: *Bag.*

to pass the noise threshold and changes of ambient light during image acquisition prevent a correct extraction of the laser projection.

Furthermore there are some deficiencies related to our system setup. As shown in figure 5.9 the laser plane continues to fan out after passing obstacles. This might results in multiple depth values for a single position, but can be easily solved by only considering the closest value.

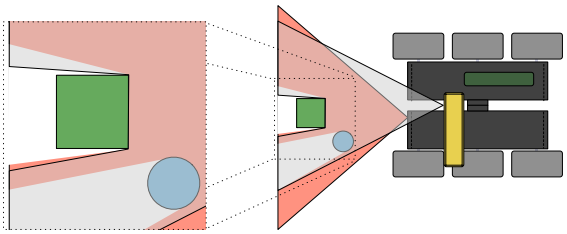


Figure 5.9: *Laser plane fanning.*

Camera and line-laser are not aligned in our system. In fact the laser is much closer to objects than the camera. The result is a cone misalignment where the angle of incident rays is steeper than the cone of vision leading to a small region nearby obstacles that is not covered by either one of the camera or the laser.

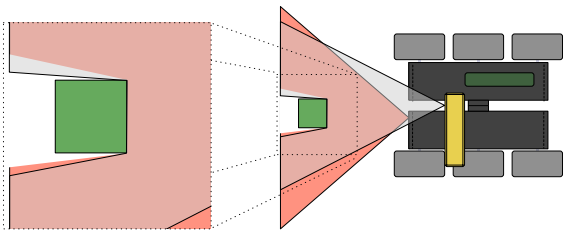


Figure 5.10: *Camera and laser cone misalignment.*

Under very unfortunate circumstances obstacles can prevent completely their detection by occluding the laser projection from the camera's field of view. This case, shown in figure 5.11, however hardly occurs.

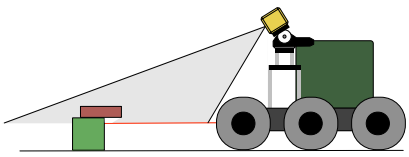


Figure 5.11: *Occlusion prevents obstacle detection.*

5.2 Lateral Drift Correction

Another major navigation specific problem we addressed by vision-based feedback, was the lateral drift of the mobile robot that occurs over time.

5.2.1 Problem Description

While navigating, our robot shows a drift as the consequence of several impact factors. The mechanical design of the robot is imperfect through the natural variability in materials. Drive motors, though having the same electrical specification, might turn at slightly different speeds under the same conditions. Additionally, external parameters like wheel traction on different surfaces, distribution of weight on the robot platform, and internal parameters like battery charge influence the robot's drive system behavior.

Thus the challenging task is to design a system that balances all parameters of the drive system in real-time.

5.2.2 Related Work

Two common approaches exist for correcting or preventing lateral drift over time: A sensor-based that relies on hardware to sense and directly correct the mechanic deficiencies, and a vision-based approach that analyzes the scene in real-time and feedbacks the results to control the hardware.

Sensor-based: The simplest sensor-based method is the experimental calibration of the robot's drive system according to external and internal parameters. Lateral drift as well as battery performance are manually measured and fed back as curve offsets into the drive controller. As the drive speed is not balanced on the fly, this method still causes drift and has to be calibrated for each environmental change.

Constantly monitoring and updating the motor speeds improves reliability and significantly reduces lateral drift. However, a complex synchronization between the different drive controllers is required; otherwise the robot might drive in a wriggly line. Furthermore, as there is no visual feedback, balancing mechanically the drive speed doesn't necessarily mean that the robot is moving at this speed.

More evolved sensors such as compasses or gyroscopes can determine in real-time the robot's heading direction. This information can be used to adjust for lateral drift, but compasses are strongly influenced by local magnetic fields, thus they are inadequate for indoor environments. Gyroscopes provide more precise heading information but nevertheless generate an accumulation error over time [Conradt, 2008].

Currently the most convincing sensor-based method is the use of optical mouse sensors [Bradshaw et al., 2007; Palacin et al., 2006]. Originally intended for high precision input devices, these optical sensors are able to observe a tiny area of the ground surface, e.g. 16×16 pixels at a very high frame rate ≥ 1500*fps*, so that even the slightest movements are detected by an optical flow algorithm. As the flow calculation is done on the sensor chip, precise movement information is directly available in real-time. Recent results show a minimal lateral drift, even over a long time. The major disadvantage of these sensors is their fixed focus that defines the distance of the robot platform to the ground surface and limits their application.

Vision-based: It was shown in [Srinivasan et al., 1996] that even less complex organisms like bees use cues derived from optical flow for navigational purposes to fly in a straight line by balancing the optical flow field information. Based on this idea, a robot should be able to navigate by optical flow. Calculation of optical flow in real-time is challenging [Camus, 1995] but can be used for navigational purposes [Tellzer, 2001]. The level of detail for calculating the optical flow in real-time is quite limited, and moreover, the environment needs to be highly textured. In realistic environments it is only possible to determine the optical flow roughly but still use this information for navigation [Sebastien, 2003].

Because of limited computational resources on low-cost robots, we decided to implement a method that locates and corrects the lateral drift over time by using distinctive feature points and the estimated focus of expansion (FOE).

5.2.3 System Setup and Visual Processing

Figure 5.12 gives an overview of our method used for determining and correcting the robot's lateral drift. Pre-processing steps follow the multi-threaded pipeline according to Figure 3.5(a) for maximal throughput.

Feature points:

Our lateral drift correction based on FOE needs good feature point detection. We implemented three different feature detectors and evaluated each for performance and detected feature quality:

KLT: Kanade-Lucas-Tomasi feature tracker [Tomasi and Kanade, 1991; Shi and Tomasi, 1994].

SIFT: Scale Invariant Feature Tracker [Lowe, 1999].

SURF: Speed Up Robust Features [Bay et al., 2008].

Our findings correspond mainly to [J. Bauer, 2007]. While the detected feature quality is slightly better with SIFT, the performance is, by a factor of 5-6, signifi-

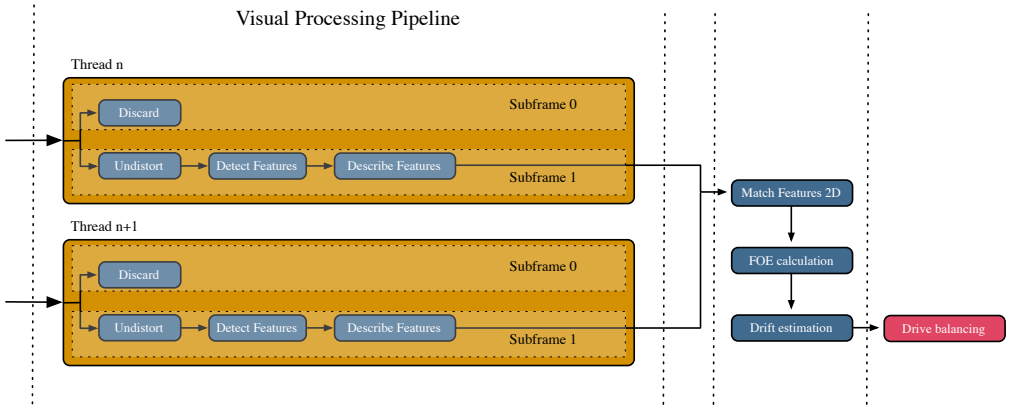


Figure 5.12: *Visual processing pipeline setup for real-time determination and correction of lateral drift.*

cantly higher with KLT and SURF. Detected feature quality is lower for KLT as it detects more features but also contains more outliers.

As the performance of this feature detectors showed to be insufficient for high-resolution images we used our own new RTSIFT feature acquisition system for fast and reliable feature detection as described in Chapter 4.

Figure 5.13 shows the scheme we are using for continuous feature detection and matching. For each sequence of n images, we detect features on the sequences first and at three other distinctive offsets (i.e. at 10, 15, 20). Feature matches are generated for three sets each involving the first image. This matching procedure results in strongly pronounced movement vectors and an additional robustness against outliers or wrong movement prediction. Outliers are easily recognized by their vector length that should fall in the range defined by the chosen distinctive image positions and their misalignment compared to its closest neighbors.

Based on the three feature-match sets, the focus of expansion is estimated and the image sequence moved to the next position $n + 1$.

Focus of expansion:

The focus of expansion is the particular point that determines the heading direction for a translational motion. It is equivalent to the epipole, a fixed point that has the same coordinates in both images. Feature points in the image sequence can only move along lines emerging from the epipole [Chen et al., 2003].

As illustrated in Figure 5.14, point p_i and p'_i must lie on the same epipolar line arising from the FOE. Thus

$$(p_i \times p'_i) \cdot v = 0 \tag{5.10}$$

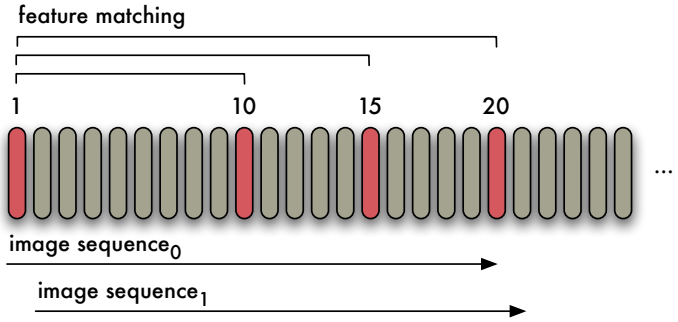


Figure 5.13: *Frame sequence feature matching*

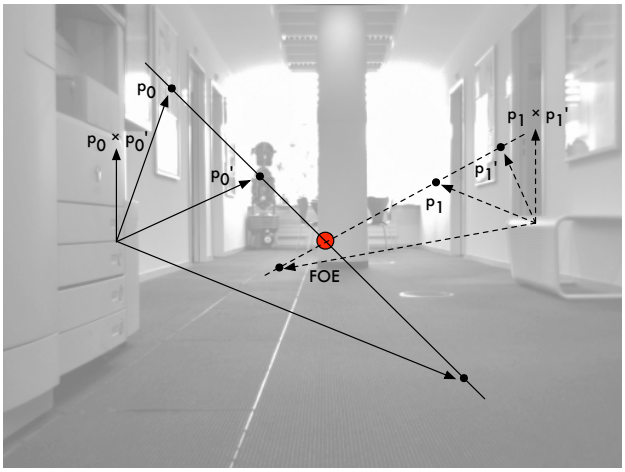


Figure 5.14: *Calculating the focus of expansion.*

while v being the focus of expansion. For each pair of corresponding points we obtain a linear equation. The FOE can be calculated from at least two equations. If the data is not exact because of noise in the point coordinates or false detected feature points, then sufficiently many pairs of matching points p_i, p_i' are needed for a good fit of the resulting equation system

$$A \rightarrow \forall_i : (p_i \times p_i') \cdot v = 0. \tag{5.11}$$

The least squares solution for v of the linear equation system A can be found by singular value decomposition (SVD) and corresponds to the last column of V in:

$$svd(A) = UDV^T \tag{5.12}$$

SVD is prone to outliers so we implemented additionally a RANSAC (RANdom Sample Consensus) extension to estimate the FOE. Results (Figure 5.16(b)), however, show a higher variance for the RANSAC approach and thereby do not justify the additional and undesired computational effort.

5.2.4 Lateral Drift Estimation and Correction

Each image of a sequence of n images results in the estimation of a FOE vector v, which should vary smoothly over time without abrupt changes between images. To remove falsely estimated FOEs we median filter the three last values of v to enforce such smoothness. Furthermore, minor variations should not result in any unmotivated correction, thus we defined a FOE-center zone in which changes will not result in an immediate drift correction. Outside a value between -1 and 1 is assigned for FOE offsets from the expected center. Hence we have an offset function

$$f(t) \in [-1, 1] \tag{5.13}$$

over time that indicates the distance of v from the expected image center and a deviation angle. Consequently, the integral over time

$$F(t) = \int_t f(t)dt \tag{5.14}$$

indicates the actual driving direction and thus its deviation from the intended straight line (Figure 5.15). If $F(t) \neq 0$ then the robot is experiencing lateral drift. Therefore, $F(t)$ can be used to balance the robot's drive system and limit lateral drift. To drive in a straight line we must keep or continuously strive for $F(t) = 0$. Note that trying to permanently make sure that $f(t) = 0$, which in theory satisfies $F(t) = 0$, in practice only causes the robot to straighten out, but will not correct for the overall deviation in driving direction after an occurrence of $f(t) \neq 0$. Using a feedback system that strives for $F(t) = 0$ can more accurately correct drift.

5.2.5 Experimental Results

We tested our method in an unmodified indoor environment. Balancing of the drive system is enabled by motor controllers that directly interact with the computer unit.

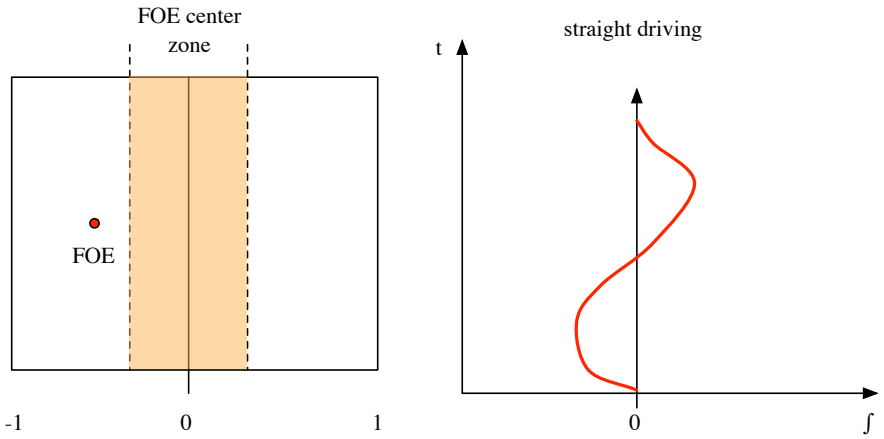
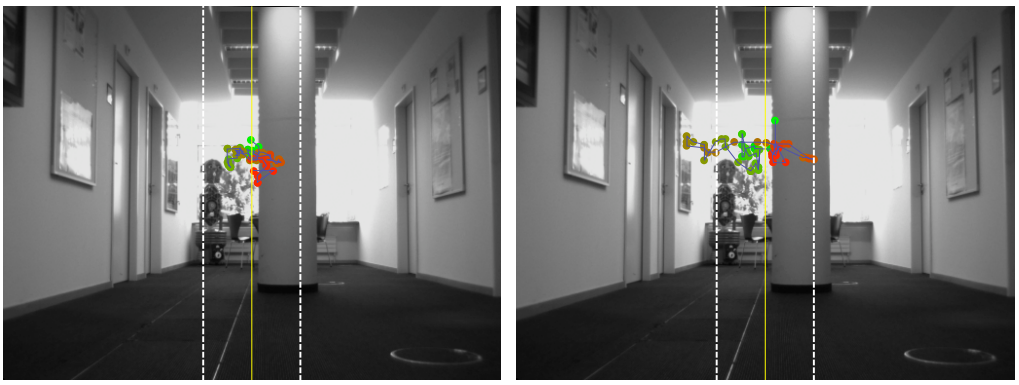


Figure 5.15: *Lateral drift estimation and correction*

Examples for FOE estimation are given in Figure 5.16(a) and 5.16(b). The image center is represented by the yellow line while the dashed white lines delimit the FOE-center zone. Estimated FOEs are shown as colored dots. A lateral drift is not recognized in case of Figure 5.16(a) as all FOEs are lying inside the FOE-center zone. This corresponds to the Graph 5.17(a). The integral $F(t)$ is 0 as only small lateral drifts occur. In Graph 5.17(b) we applied intentionally an unbalanced speed to the robots drive system, resulting in a lateral drift to the left side. The graph illustrates this drift and the integral. Finally, we applied our lateral drift correction method as seen in Graph 5.17(c). The robot's drive system is configured in the same way as before but as soon as the drift is visually recognized, the robot starts to rebalance its drive system and manages to correct nearly the initial lateral drift. However, balancing the robot's drive system properly has shown to be a challenging problem during our experiments, as slight oversteering can occur which may result in driving in a wriggly line.

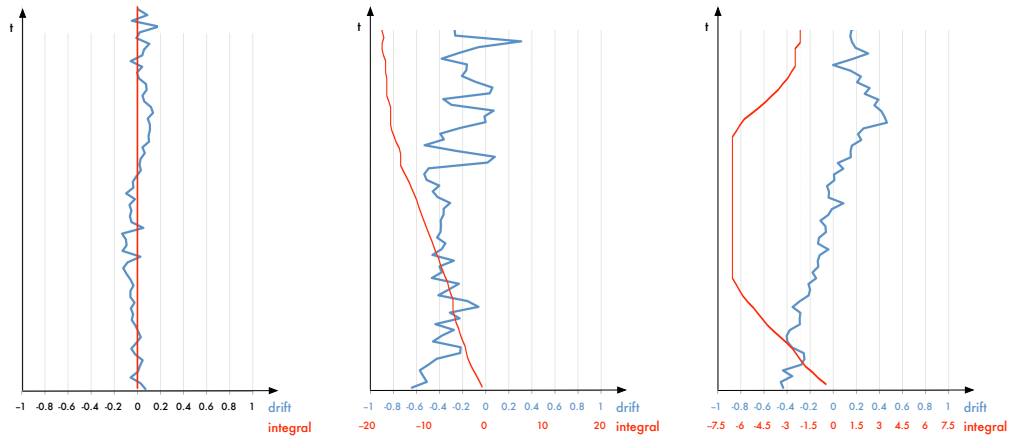
5.3 Localization

A key component of semi-autonomous robot navigation is localization. Localization during navigation provides a regulatory feedback for the robot's drive system and a response to user defined navigation tasks. Moreover, a close approximation of the robot's position and orientation inside the environment simplify the mapping of features and obstacles (Chapter 6).



(a) (b)

Figure 5.16: *FOE estimation while driving straight. (a) SVD (b) RANSAC*



(a) (b) (c)

Figure 5.17: *Lateral drift and accumulated integral*

5.3.1 Problem Description

Our mobile robot platform is not equipped with any positioning sensors. Thereby, localization is restricted to vision through the Bumblebee 2 stereo camera. Vision-based localization in an unknown indoor environment represents a challenging task. Features are sparsely distributed and tend to be unstable, being only valid for a short period. However, translational and rotational changes of the robot's position or orientation need to be determined as accurate as possible, since they are critical for navigation.

5.3.2 Related Work

To determine changes in the position and orientation, various sensors and systems exist. Most commonly used are optical encoders and gyroscopes.

An optical encoder includes an optical sensor and a special reflector to measure the wheel rotation. From this rotation the values for velocity, acceleration and displacement are calculated. Optical encoders have a limited accuracy that is given by the number of encoder steps on the reflector. The wheel rotation though does not necessarily have to reflect the robot movement [Borenstein et al., 1996].

Gyroscopes provide angular velocity rate information, which corresponds to rotation around a specific axis. Precise orientation estimates are not possible, because each measurement contains a drift rate that increases over time [Barshan and Durrant-Whyte, 1994]. In addition, depending on the surface, wheel friction can make the angular velocity erratic.

Due to their inaccuracy both sensors are in many cases combined with a re-localization, e.g. landmark correction, to reset the accumulated error.

Landmarks are as well the main component of indoor localization systems. These systems depend on markers that are placed throughout the environment. At least one marker needs to be visible at all times. Hence they are mainly applied to the ceiling where occlusions are less likely to occur.

Commercial indoor localization systems [EvolutionRobotics, 2005], [Hagisonic, 2008] use infrared-responsive or active markers, so called beacons. Infrared-responsive markers are illuminated by infrared light. A camera that is responsive in the infrared spectrum recognizes the reflection and the system calculates the robot's relative position from the markers id, size and image position. The typical precision such a system achieves is between $1 - 2^{\circ}$ for angular orientation and $2 - 4cm$ for the position. Landmark placement is necessary every $2m$ and the distance of the robot should be between $2.5 - 5m$ from the marker. Beacons, as active markers, have the advantage that they actively send out information in several directions and can answer to requests. This simplifies the marker identification. However, these indoor localization systems constrain the robots operation

to artificial prepared environments.

Vision-based approaches have the advantage, that they can exploit natural landmarks instead of pre-defined markers. Unfortunately this leaves the whole image processing, from the recognition of landmarks to the position calculation, to the robotic platform.

A vast amount of work exists for the vision-based robot localization using natural features. This includes feature based localization using SIFT [Se et al., 2002], the simultaneous localization and mapping with monocular vision [Davison et al., 2007] and stereo vision [Karlsson et al., 2005]. In combination with odometry information, environment models and probabilistic filters the robot's position and orientation can be estimated quite accurately. The main disadvantage of these approaches is the dependence on a highly textured environment.

We propose a method for robot localization in sparsely textured environments that accurately estimates the position and orientation of the robot platform even in the presence of unstable features. Our method is based on corresponding 2D features and considers translations and rotations as independent movements. Errors introduced by this approximation can be corrected with re-localization. Results show an accuracy that is comparable to commercial indoor localization systems.

5.3.3 System Setup and Visual Processing

The visual processing pipeline is configured according to figure 4.2. We detect and describe features in continuous images using RTSIFT. This allows us to obtain 2D and 3D feature correspondences.

[Shen et al., 2006] reported good results for the localization based on 3D features. Though in our system we will only be able to locate the feature position up to a particular accuracy, typically given by the spatial sampling of the image.

In a standard perspective stereo setup (Figure 5.18) we define the ideal distance of a feature p_0, p_1 to the lens by

$$z = \frac{b \cdot f}{d} \tag{5.15}$$

Due to the discretization error involved in measuring the pixel disparity d, we will have an uncertainty in the depth measurement z. This depth uncertainty is proportional to the square of its related depth value [Chai and Shum, 2000].

$$\Delta z = \frac{z^2}{b \cdot f} \cdot \delta x \tag{5.16}$$

Figure 5.19 illustrates the theoretical error versus depth value for two different baselines. The Bumblebee 2 camera has a baseline of $b = 120$mm. Increasing this baseline reduces the depth error, but also the area covered by both cameras.

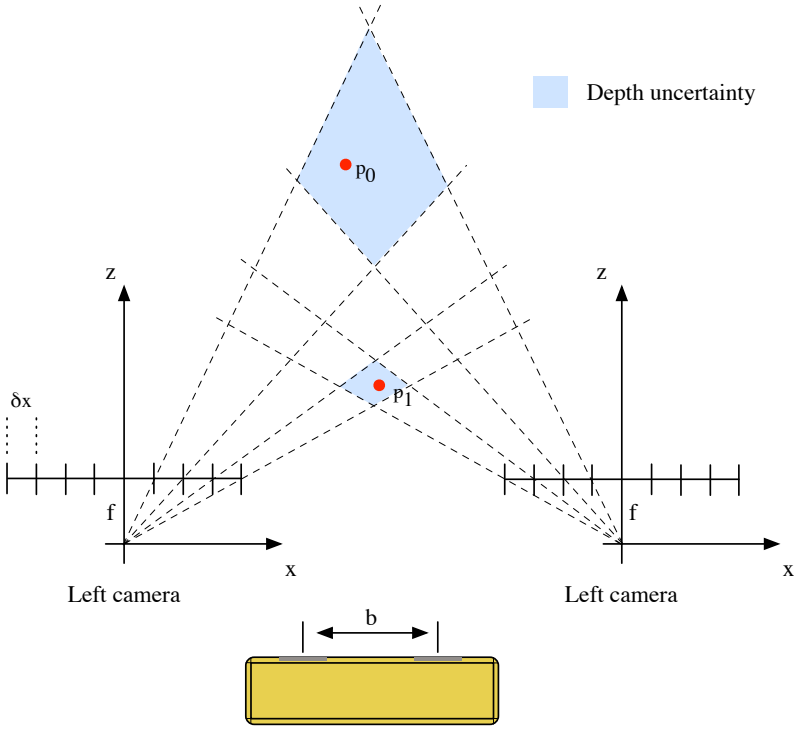


Figure 5.18: *Perspective stereo setup.*

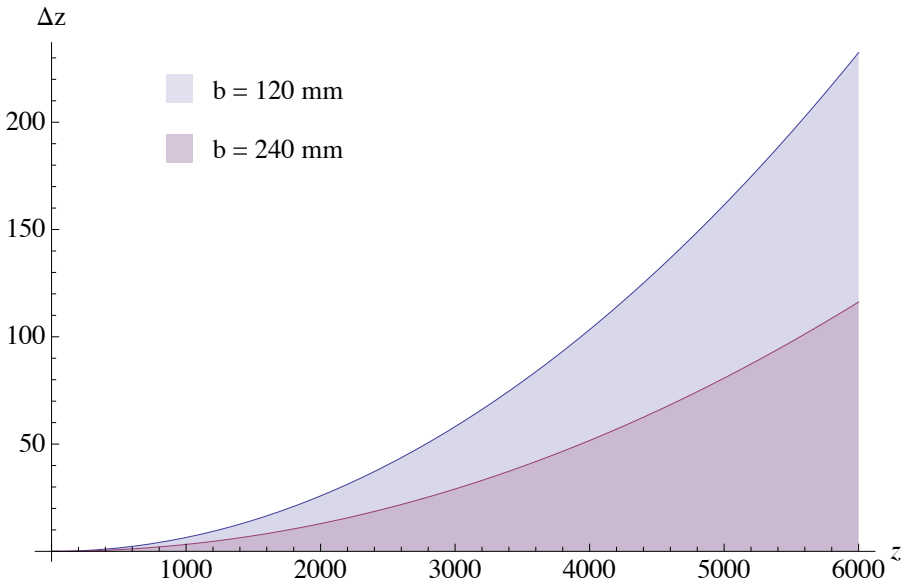


Figure 5.19: *Depth error for different baselines and distances.*

Therefore the baseline is limited and depends on the required feature correspondences.

Feature correspondence is difficult to achieve in a sparsely textured environment, especially if the features are unstable. Sufficient corresponding features can only be found between consecutive images. Localization is limited to a local image-to-image estimation. Actual changes are quite small compared to the depth error that accumulates over time. In our environment, features are usually $\geq$ 2m away from the robot platform. The maximal depth error for an image-to-image estimation is twice $\Delta z \approx 50$mm (Figure 5.19) which is greater than the actual inter-image translation. Furthermore the Bumblebee 2 camera is oriented parallel to the robot's driving direction resulting in small pronounced movement vectors with high uncertainty.

Estimating the translation:

To solve these problems for our mobile robot platform we determine the translation based on 2D features. They are acquired perpendicular to the movement direction by the Apple iSight camera module. In particular we are using features on the ceiling. The advantage is pronounced movement vectors. Moreover, we are not accumulating anymore depth uncertainties, because the distance to observed features stays constant. The translation of the robot platform can be directly determined from the translation of 2D feature points (Figure 5.20).

As features on the ceiling have approximately the same distance with only minor variations, we can define a single value z_M according our indoor environment for simplification. Non-linear Levenberg-Marquardt optimization (Equation 5.18) was used to determine the optimal translation t in the least-square sense.

$$t = \frac{z_M}{f} \cdot (p'_i - p_i) \tag{5.17}$$

$$E(t) = \sum_{n=0}^{N} [p'_i - f(p_i, t)]^2 \tag{5.18}$$

Alternatively the value z_M could be determined from the feature p_i. This showed to be problematic, as due too insufficient texture stereo correspondences could not always be established. Translational movements are bound to a single axis as the robot platform can not move sideways. For directional changes the mobile robot platform has to rotate. Rotations are handled completely independent from translations, which contributes to a more accurate localization.

Estimating the rotation:

To determine rotations we considered the method proposed in section 4.9 for feature integration. Instead of rotating the camera, it is kept stable and the robot rotated. Unfortunately this method failed. While rotating the robot platform acquired images have a high motion blur that prevents the detection of any feature.

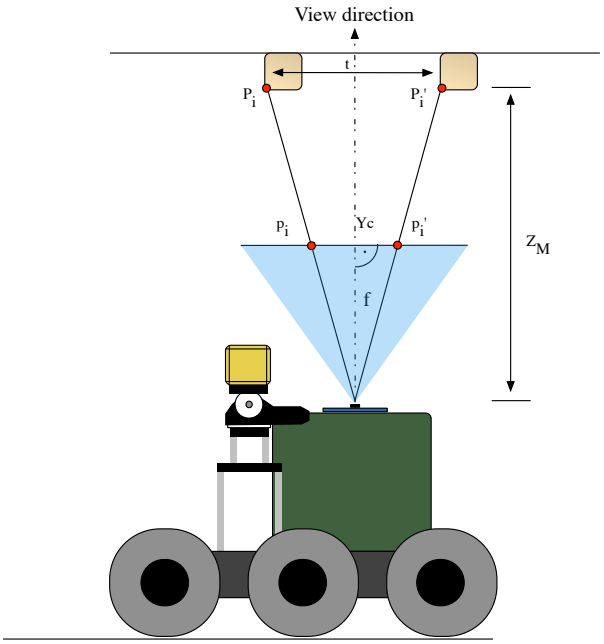


Figure 5.20: *Determining translational movement of robot platform based on 2D features.*

High wheel traction is responsible for this case. The front- or back-most wheels are dragged over the ground. Depending on the ground texture this requires a certain speed that is too high compared to the Bumblebee's 2 frame acquisition rate.

For this reason we choose a different approach. Our solution is as follows: Before actually rotating the robot platform, a single image is acquired. We start to rotate the robot in small intervals. Between each interval the robot is stopped and the camera rotated in the opposite direction. During the camera's rotation we subtract the current from the pre-acquired image and determine the rotation angle by our original feature integration method (q.v. Chapter 4). The angle that corresponds to the image with the smallest difference corresponds to the rotation of the robot platform. Afterwards the camera is set to its original position. These steps are repeated and the robots drive system adjusted until the desired rotation is achieved (Figure 5.21).

5.3.4 Experimental Results

Figure 5.22(a) shows the experimental results of our distance estimation. To measure the accuracy of our method the robots drive system was turned on for differ-

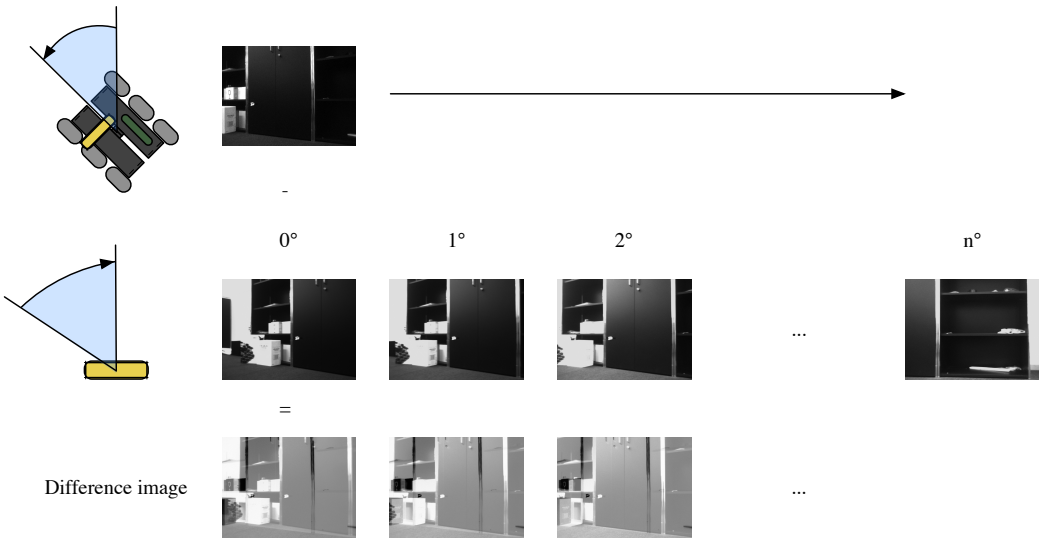


Figure 5.21: *Determining rotational movement of robot platform.*

	4s	8s	16s	32s
Distance	683mm	1409mm	2934mm	5889mm
Estimation	642mm	1360mm	2912mm	5878mm
Error	41mm	49mm	22mm	11mm

(a)

	45°	90°	180°
Rotation	45°	95°	183°
Estimation	44.8°	96.6°	185°
Error	0.2°	1.6°	2.5°

(b)

Figure 5.22: *a) Accuracy of distance estimation. b) Accuracy of rotation estimation.*

ent periods (4s, 8s, 16s, 32s). The height of the ceiling was determined and set to $z_M = 2300mm$. After the robot stopped we measured the traveled distance with a laser-ranger and compared it with our vision-based estimation. For distances between $700mm$ and $6000mm$ the translational error is $\leq 50mm$, which is below the depth error resulting from a single 3D feature with $z \geq 2000mm$.

For testing the accuracy of the vision-based rotation estimation the robot was rotated in discreet time steps aiming for three different rotations (45°, 90°, 180°).

Mechanical deficiencies of the robotic system and the alternating wheel friction lead to a different real rotation (Figure 5.22(b)). The estimated angle has an error of $\leq 2.5°$. It is increasing with the rotation angle, as more discreet steps are needed whereby the error accumulates.

CHAPTER 6

ENVIRONMENT MAPPING

Environment mapping is the process of building a map using information gathered by the robot sensors. This map can be used for navigation and visualization. Our research focused on vision-based indoor environment mapping. Sparsely textured surfaces, unstable features, sudden illumination changes, moving objects and the restricted computational power of our low-cost mobile robot platform demand a novel approach to generate a reasonable map representation.

6.1 Problem description

We are using two vision-sensors to obtain information about the environment. A Bumblebee 2 stereo camera for acquiring 2D- and 3D features, as well as an Apple iSight camera module for localization. 3D features need to be integrated from different camera positions. Either at a single viewpoint when the camera is rotating or from different viewpoints throughout the environment. To integrate features at a single viewpoint we employ our proposed feature acquisition and integration (Chapter 4). The angular precision of this feature integration is limited, which is why features might be represented multiple times in the resulting point cloud at slightly shifted positions. A similar problem arises for the integration of point clouds from different viewpoints. Spatially close features might represent the same real world feature but are projected to different positions. Their position varies in accordance with the robot's distance to the feature at acquisition time and the localization precision.

6.2 Related work

Several well established algorithms exist for the integration of 3D features into a single representation. The most important in the area are Iterative Closest Point (ICP) [Besl and McKay, 1992], Least-square fitting (LSF) [Arun et al., 1987] and sparse bundle adjustment (SBA) [Triggs et al., 2000].

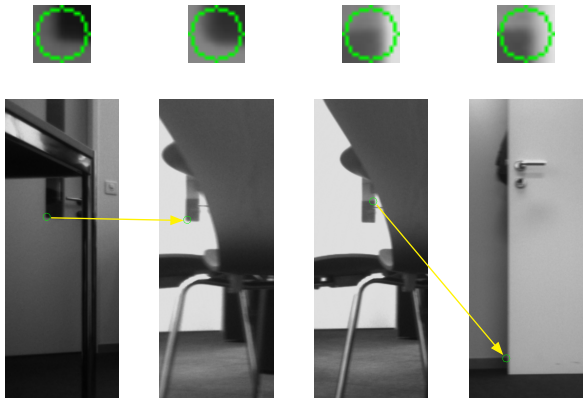
ICP is an algorithm that is often used to reconstruct 3D surfaces or localize the robot's position from different dense point clouds acquired by a laser scanner. Therefore it associates points by the nearest neighbor criteria. The transformation (translation, rotation) between the point clouds is iteratively revised to minimize the inter-point distance. ICP is very simple and commonly used in real-time. Nevertheless ICP requires dense point clouds that overlap to a large extend for determining a correct transformation, while our 3D features are sparsely distributed and have only little overlap.

LSF determines the transformation based on singular value decomposition (SVD), decoupling the translation and rotation. Two corresponding 3D point sets are needed. The reliability of the transformation estimation increases with the number of corresponding points. In combination with RANSAC the influence of outliers can be reduced iteratively. Unfortunately the number of reliable 3D features correspondences we achieve in our environment is too small compared to the number of possible outliers for using this approach.

SBA simultaneously refines the 3D feature positions describing the environment as well as the parameters of the relative motion and the optical characteristics of the camera. This is done according to an optimality criterion involving the 2D projections of the features. A set of 3D features and its corresponding 2D projections are required. Unstable features prevent in our case the tracking of features and thereby locating the 2D projections.

The SIFT feature descriptor (Section 4.7) showed to be insufficient for establishing feature correspondences under viewpoint changes. Figure 6.1 emphasizes this problem that occurs in indoor environments with sparsely textured surfaces. Feature detection is basically limited to edges. These edges are part of structures that repeat throughout the environment. The feature descriptor is indistinguishable between these structures and a unique assignment therefore impossible.

Given these preconditions, none of the standard algorithms is able to integrate the 3D features acquired. Hence we propose a system that relies on vision-based localization to integrate 3D features from different camera positions. Our system is supported by stereo vision and active laser illumination for generating a dense environment representation. The environment mapping is semi-automatic and requires some limited human assistance.



(a) (b)

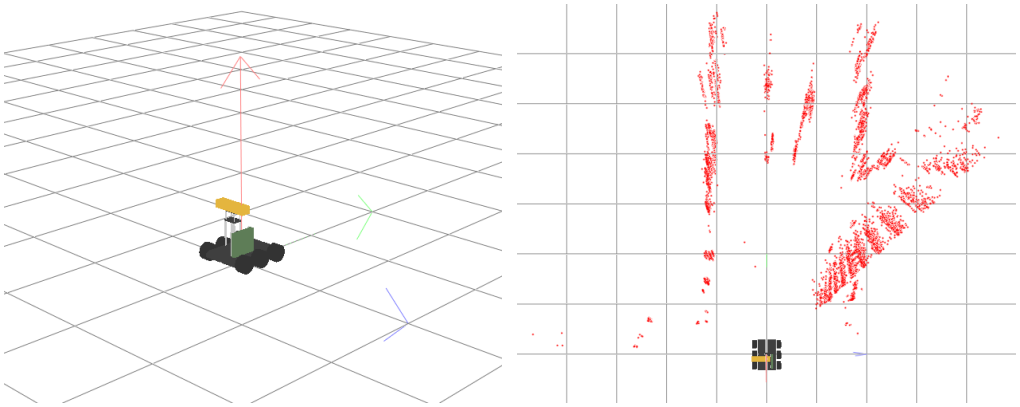
Figure 6.1: *Ambiguity for establishing feature correspondences from different viewpoints. (a) Feature descriptors match, correct correspondence is established. (b) Feature descriptors match, but feature belongs to different structure.*

6.3 System Setup and Visual Processing

To generate a map representation of the environment the mobile robot platform is placed at an arbitrary position. This initial position will be considered as the maps center position (Figure 6.2(a)). The robot has no prior knowledge about the environment. We start to acquire 2D- and 3D- features in a 130° wide area (Section 4.3). Camera rotation is retrieved from 2D feature correspondences. 3D features are integrated using the angular information and projected into the maps world coordinate system (Figure 6.2(b)).

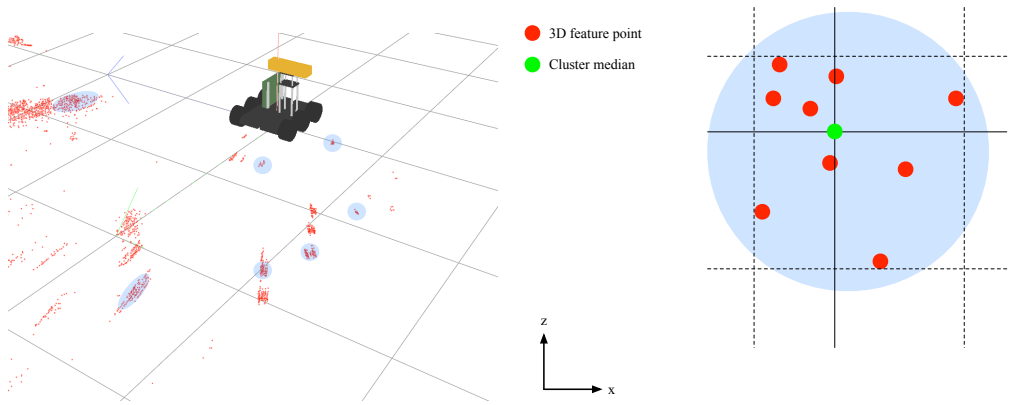
3D features with high depth uncertainty are excluded from the map. We defined the threshold at $6m$ which is equivalent to a depth uncertainty of $\approx 230mm$ (Figure 5.19). While rotating the camera, duplicate 3D features might be acquired. They build clusters in the map (Figure 6.3(a)). Features inside clusters are integrated by comparing their descriptors and using the depth uncertainty information. A median operation merges similar features that are placed inside the same depth uncertainty area (Figure 6.3(b)). In addition we assign a reference count to the resulting 3D feature. Features with numerous references represent good landmarks and are visualized accordingly in the environment map.

The user can navigate the mobile robot platform through the environment. As the environment exploration is semi-automatic, navigation is limited to the definition of an exploration objective. During navigation the robot is self-localizing its position (Section 5.3) and correcting the drive system's drift (Section 5.2). Ob-



(a) (b)

Figure 6.2: *Environment mapping. (a) Initial position of the mobile robot platform is considered as the center of the environment map. (b) Single 3D feature scan.*



(a) (b)

Figure 6.3: *Feature clustering. (a) Possible duplicate 3D features building clusters. (b) Feature merging with median operation in XZ-plane.*

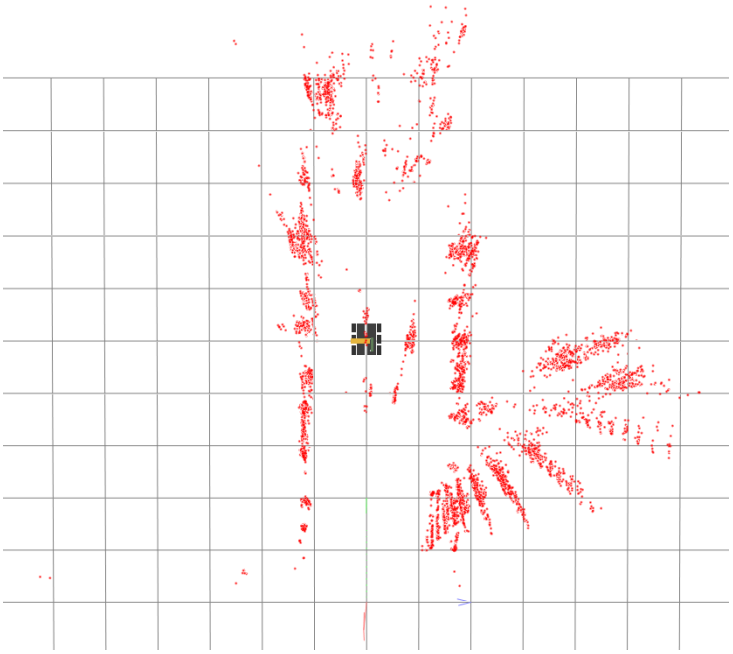


Figure 6.4: *Environment map after scans from multiple positions.*

stacles are detected and avoided (Section 5.1). After reaching a new position, the feature acquisition and integration stage is repeated. Localization information integrates new features to extend the environment map (Figure 6.4).

If insufficient 3D features are found, stereo-vision and active-illumination can be employed to add depth values from edges and homogeneous areas to the environment map.

Stereo-vision: In stereo-vision, image structures are correlated. The disparity between the structures stereo projections is directly related to the distance of the object they represent.

We developed an algorithm that performs the calculation of the disparity map at interactive frame rates on high resolution images. Figure 6.5 gives an overview of our stereo system.

Captured frames are pre-processed and converted to grayscale images (Figure 3.2(a)). Inside the visual processing pipeline, frames are undistorted using the internal camera parameters to satisfy the epipolar constraint [Hartley and Zisserman, 2004]. Our stereo matching algorithm will calculate subsequently the disparity map with correspondence analysis.

Correspondence analysis tries to solve the problem of finding the correlation of pixels in-between two stereo frames. We employ a region based approach that

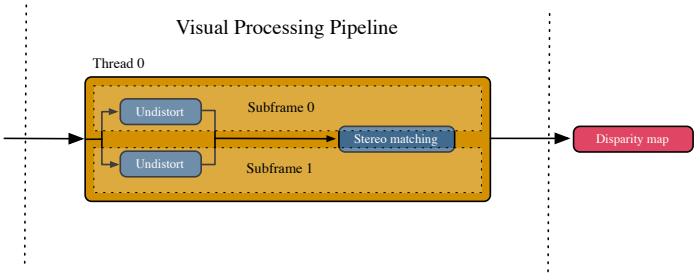


Figure 6.5: *Disparity map calculation.*

solves the correspondence problem for every single pixel in the image. A block consisting of the middle pixel and its surrounding neighbors is taken from one stereo frame and matched to the best corresponding block in the second stereo frame. Selecting the optimal block size is difficult. Large blocks increase the correctness of the disparity map but also the processing time. We found a block size of 9×9 to represent a good compromise.

The sum of absolute differences algorithm (SAD) computes the intensity differences for each pixel (x, y) inside an $n \times m$ window (Equation 6.1).

$$SAD(x,y) = \sum_{j=-\frac{m-1}{2}}^{\frac{m-1}{2}} \sum_{i=-\frac{n-1}{2}}^{\frac{n-1}{2}} |g_0(x+i, y+j) - g_1(x+i, y+j)| \tag{6.1}$$

The minimum difference over the epipolar line is chosen to be the best match, with the disparity being the actual horizontal pixel difference. To improve the quality and runtime of the algorithm several matching constraints are implemented.

Uniqueness constraint: Each block has a sole match. Blocks with multiple matches are considered as outliers and no disparity value is assigned.

Cross correspondence constraint: A two-way matching is implemented. The search for the corresponding block is performed from the left to the right image and in the opposite sense. This cross matching doubles the processing time, but efficiently eliminates outliers as correspondences are only established if the same disparity is found independent of the search direction.

Area constraint: The search area is limited to a minimal and a maximal disparity. Both values depend on the environment and the range inside which structures are expected.

Given the disparity map and the geometry of our stereo setup, the 3D position of points in the image is computed with triangulation. The optical axes of the two cameras in our stereo setup are parallel. Their lens centers are horizontally

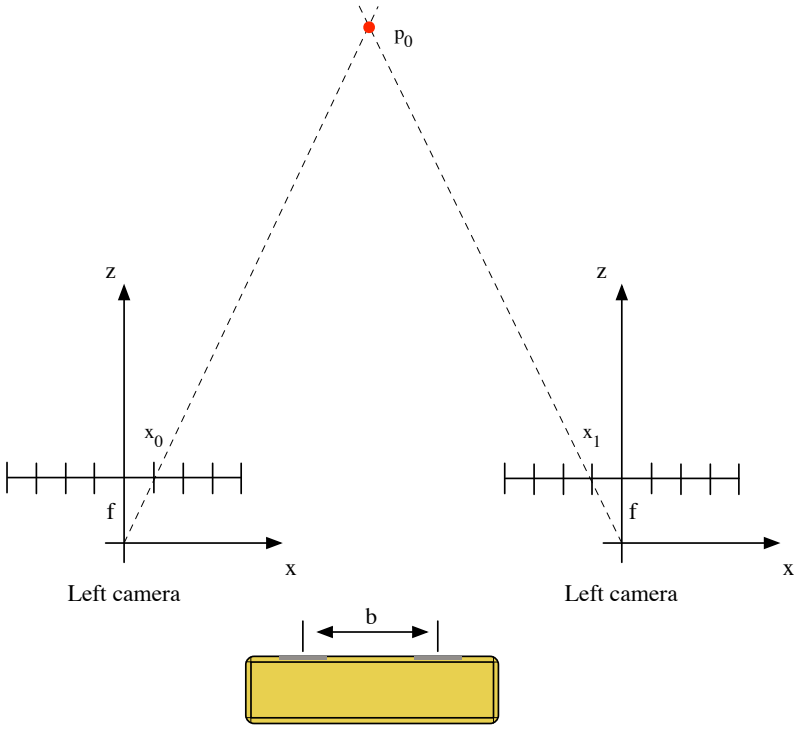


Figure 6.6: *Triangulation according the standard camera model.*

separated by the baseline b. The baseline is perpendicular to the optical axes. f is the focal length of both cameras. XZ is the plane where the optical axes lie, the XY plane is parallel to the image plane of both cameras. The origin of the world reference system is the lens center of the left camera. This setup is shown in Figure 6.6 and called the standard camera model. The 3D position (x, y, z) of a point p_i, can be reconstructed from the perspective projection of p_i on the image planes (x_0, x_1, y_0) of the cameras.

$$d = x_1 - x_0, \quad z = \frac{b \cdot f}{d} \tag{6.2}$$

$$x = \frac{x_0 \cdot z}{f}, \quad y = \frac{y_0 \cdot z}{f} \tag{6.3}$$

Laser scanning: To acquire a dense surface representation of the environment we apply active-illumination. The Bumblebee 2 stereo camera is used in combination with a pulsed $532nm$ line-laser. This wavelength shows the highest

spectral response on the cameras imaging sensor. Therefore detection should be straightforward, even on large distances and under difficult lightening conditions.

The laser projects a vertical line and provides the required structure to obtain depth values from homogeneous areas. Continuously the laser plane projection is extracted from both stereo images. Extraction of laser line uses the image subtraction method that synchronizes the image acquisition with the laser pulses. The energy spectrum of the laser projection corresponds to a gaussian curve. We determine the laser peak $\hat{X}$ with linear approximation [Forest et al., 2004] according Equation 6.4.

$$\hat{X} = x_0 - \frac{y_0 \cdot (x_1 - x_0)}{y_1 - y_0} \tag{6.4}$$

The disparity between the laser-line projections gives us the distance of the observed surface. As the laser can be automatically deflected within a $60°$ angle (Figure 2.26), a dense representation of the surface is acquired. The precision depends on the distance of the mobile robot platform. Due too the limited angular resolution of the deflection mechanism and especially the depth uncertainty (Figure 5.18) surfaces will be represented more rough with increasing distance.

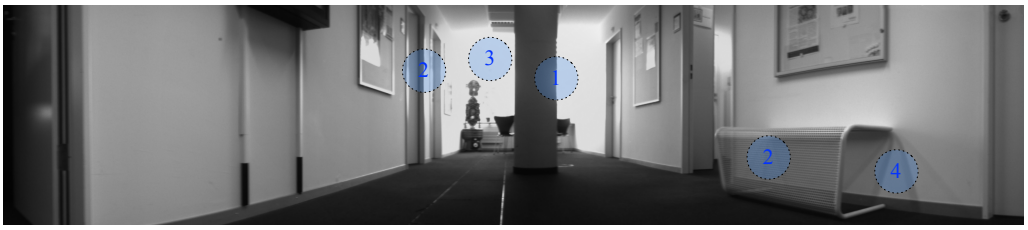
6.4 Experimental Results

We placed our mobile robot platform inside an unmodified indoor environment. 2D- and 3D feature points are acquired within a $130°$ angle (Figure 6.7(a)) at 5 different positions ($z = 0, z = 1m, z = 2m, z = 3m, z = 5m$). At each position, 3D feature points are automatically integrated. The top view of the resulting environment map is shown in Figure 6.7(b). An approximate representation is generated, although features are too sparsely distributed to allow a realistic perspective view (Figure 6.7(c)).

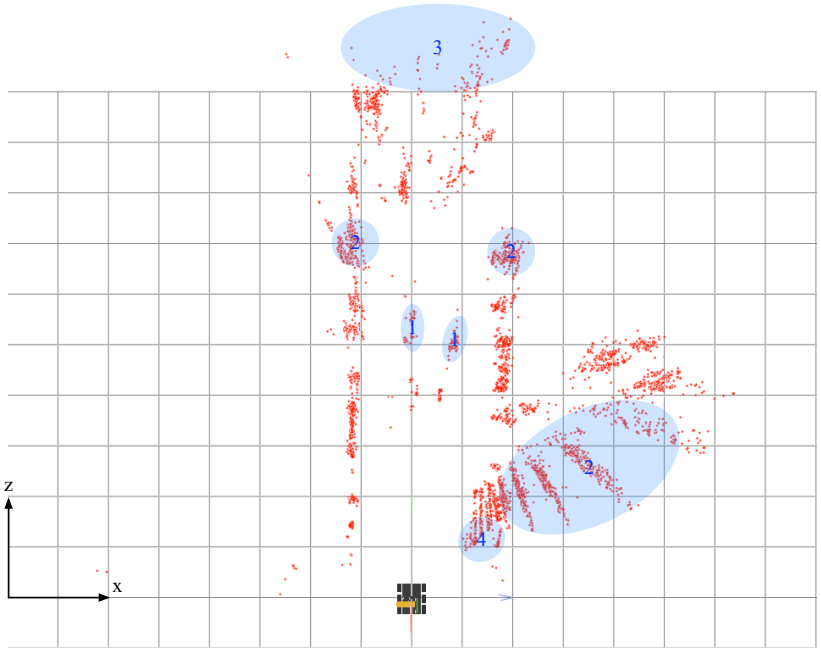
Feature based environment mapping has certain limitations (Figures 6.7(a) and 6.7(b)). Features residing on object borders move relative to the robot position and cause wrong depth estimates (1). Structures with similar appearance or reoccurring patterns (2) prevent the unique identification of features. Assigned depth values are therefore random. The limited dynamic range of the Bumblebee 2 camera can lead to overexposed image regions, especially near windows (3). These regions are error-prone and features detected within not reliable. Shadows (4) depend on the environment illumination and the robot position. Features are reliable if neither the robots position nor the illumination changes. Reflections and transparent surfaces might introduce additional errors. Despite these problems, environment mapping based on 3D features has the advantage of achieving a reasonable environment representation in real-time on the mobile robot platform.

We repeated the environment mapping employing stereo-vision. Figure 6.8(a) shows the panoramic disparity map. Disparity values are limited to edges through the stereo block matching algorithm. The top view of the reconstructed environment map (Figure 6.8(b)) is comparable to the feature based environment mapping, though map features are denser and depth values have a higher variance. Environment mapping with stereo vision shows the same limitations as feature based mapping (1, 2, 3). The uniqueness constraint of the stereo block matching algorithm contributes significantly in reducing outliers from structure similarities (2). Acquiring dense depth values along edges produces a reasonable perspective view (Figure 6.8(c)). Unfortunately stereo vision is computationally expensive. In addition each disparity map creates approximately 100.000 points. This makes the environment mapping with stereo vision and the view integration inappropriate on the mobile robot platform.

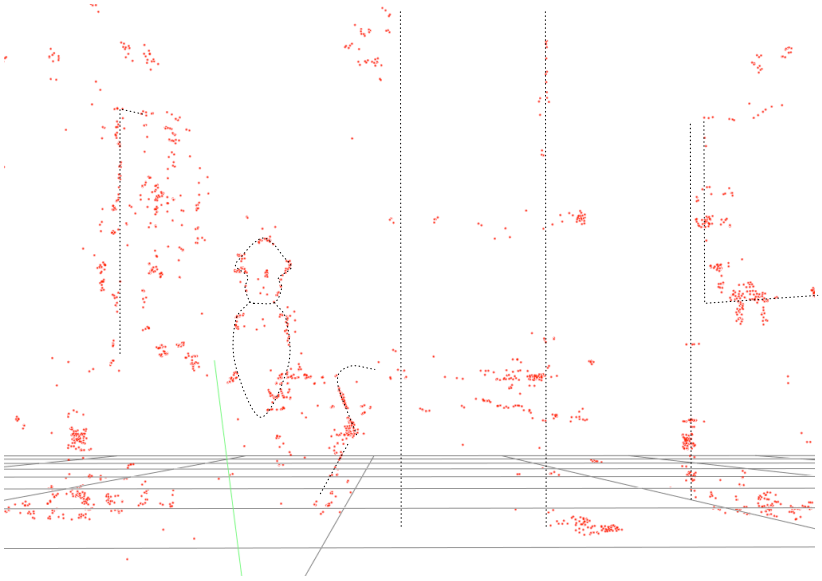
Finally we used laser scanning to obtain a close range map of selected indoor areas (Figure 6.9(a)). Extraction of the laser stripe showed to be difficult on large distances and under normal lightening conditions. The resulting map contains therefore only a few points (Figure 6.9(b)). An appropriate environment representation is not obtained. In contrast we observed excellent results for scans in the range of $1 - 2m$. Figures 6.9(d), 6.9(c) and 6.9(e) show details from objects in the environment. Even structures with reoccurring patterns can be acquired (Figure 6.9(f)). Close range scans have the advantage of a reduced depth uncertainty ($6mm - 25mm$). The precision of the scanning is mainly limited by the laser stripe segmentation. Errors in the segmentation highly influence the result and represent a major source for outliers. Laser scanning uses only basic image processing operations and thus can be done in real-time.



(a)

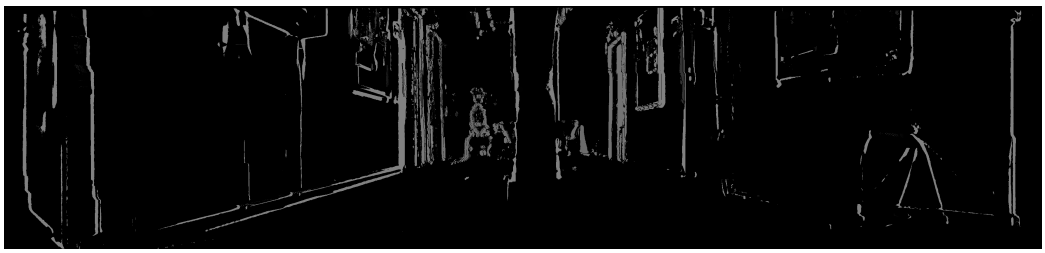


(b)

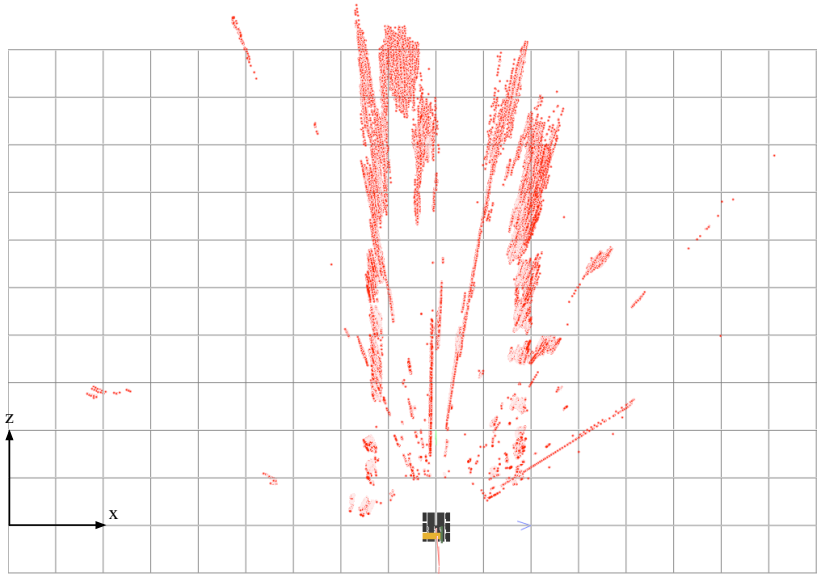


(c)

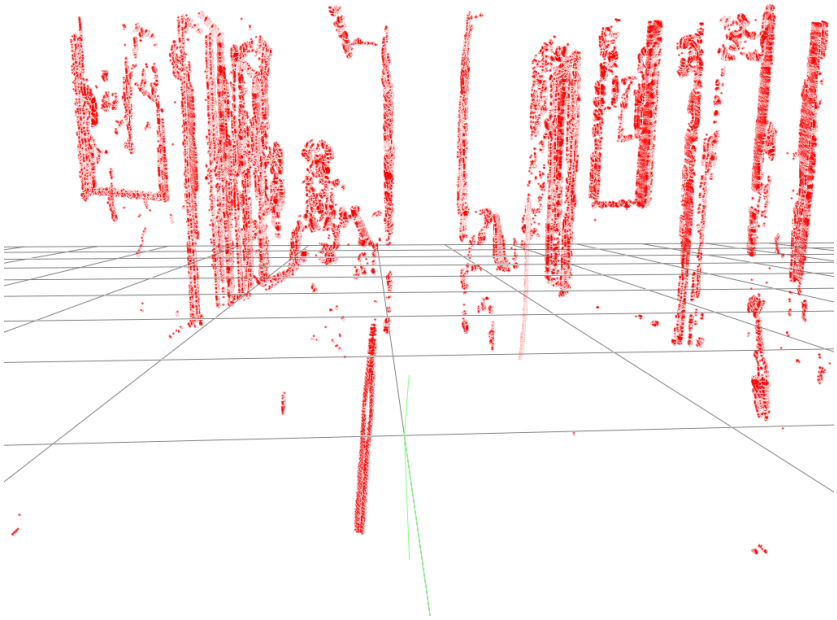
Figure 6.7: *Environment mapping with 3D feature points. (a) Panoramic view reconstructed from images acquired within a 130° angle. (b) Top view. (c) Perspective view.*



(a)

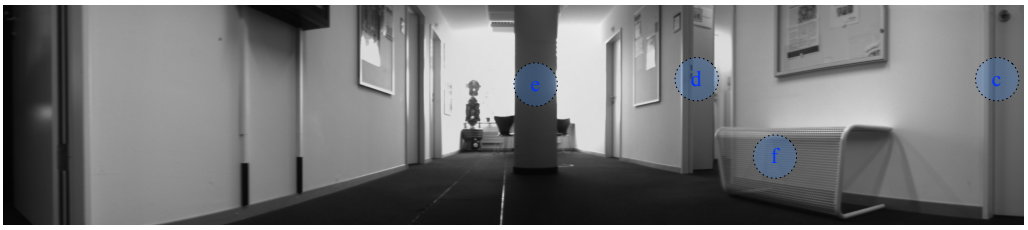


(b)

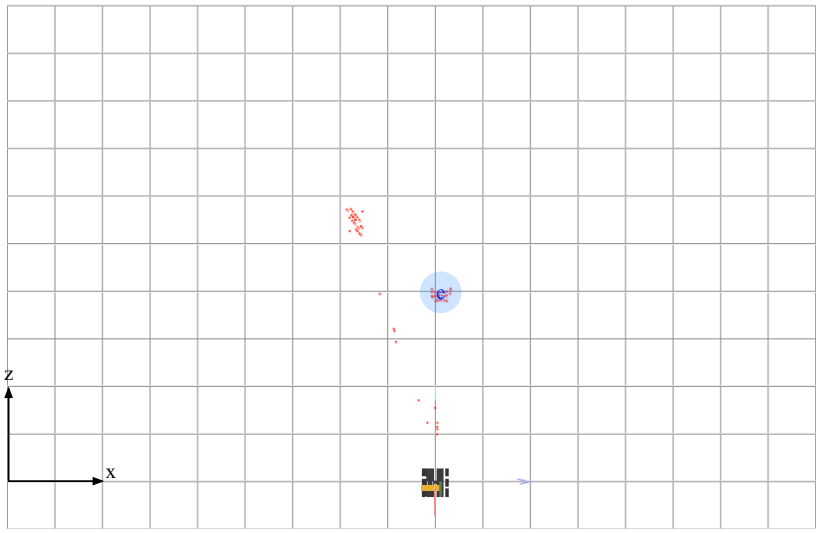


(c)

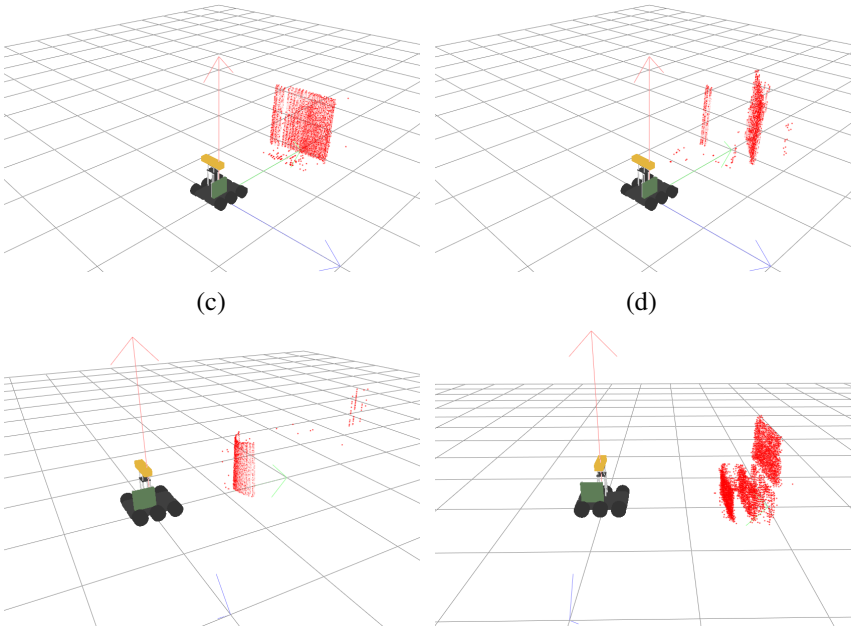
Figure 6.8: *Environment mapping with 3D stereo. (a) Panoramic view reconstructed from disparity images acquired within a 130° angle. (b) Top view. (c) Perspective view.*



(a)



(b)



(c) (d)

(e) (f)

Figure 6.9: *Environment mapping with active-illumination. (a) Panoramic view reconstructed reconstructed from images acquired within a 130° angle. (b) Top view. (c)-(f) Structure details.*

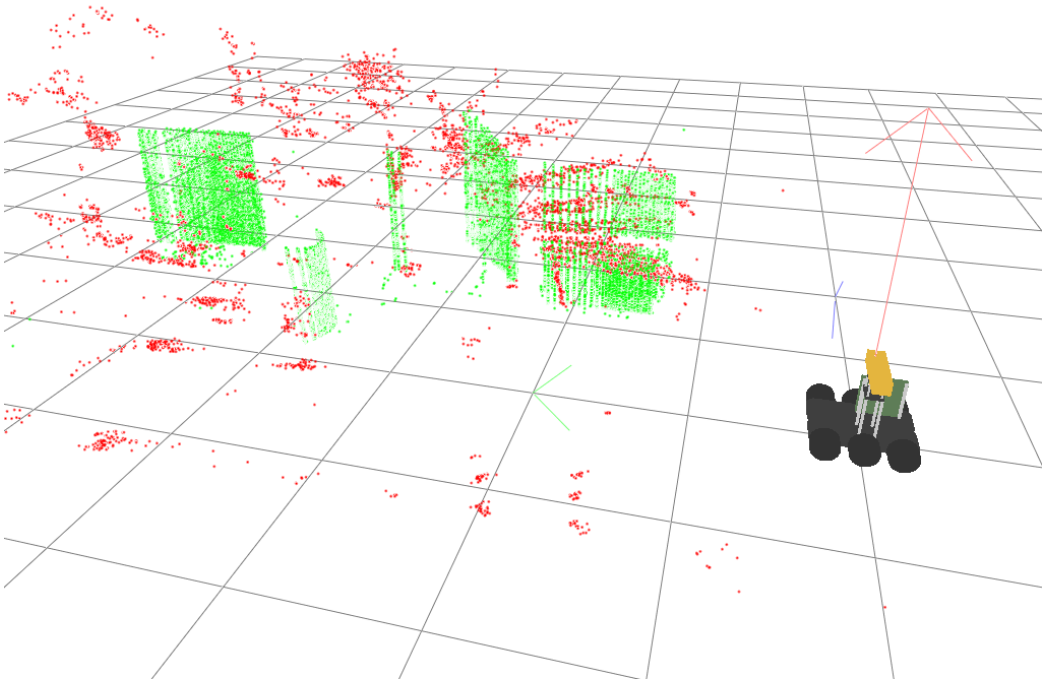


Figure 6.10: *Environment mapping with features and active-illumination.*

CHAPTER 7

CONCLUSIONS

7.1 Summary

In this thesis, a new real-time vision-based mobile robot approach for indoor environment mapping is presented. By incorporating commercial off-the-shelf components, we demonstrate that a flexible and robust mobile robot system can be constructed at low-cost. Additionally our system provides an unmatched independent operation time essential for extended environment exploration.

We show that the environment perception can be based solely on vision, even if the available computational resources are limited. Due to our fast multi-threaded pipeline approach and novel algorithms, visual data can be processed and analyzed in real-time. Specialized signal processing hardware or off-platform processing is not required.

We present an efficient algorithm for real-time feature acquisition and integration. Our algorithm is able to detect features with a high degree of reproducibility in unstructured environments. The feature detection is insusceptible to brightness and illumination changes. We combine it with a modified SIFT-based stable scale-invariant feature description. Our modification leads to a significant performance improvement. Utilizing spatial and temporal coherence reduces outliers substantially. Our proposed feature integration is optimized for indoor environments that lack densely textured features and provides accurate results at comparatively low computational cost. No other previously proposed method has been capable of that.

We introduce a regulatory vision-based feedback system that adapts the robot behavior to the unknown environment. Two major issues are addressed: The detection of obstacles and the detection and correction of lateral drift.

Our obstacle detection system combines coarse sonar-based obstacle detection with a vision-based refinement under active-illumination. We have the advantage of real-time detection along with a precise definition of the obstacles position and shape. The vision-based refinement is performed on demand in real-time. Obstacles are reliably detected, independent of their texture and shape. Furthermore, the precision of our algorithm is advantageous for selecting the correct obstacle avoidance strategy.

We propose a method for real-time vision-based lateral drift correction. This method reduces the effect of external and internal parameters that influence the robot drive system. Our method makes the drive system independent of the environment and technical imperfections, as all parameters are balanced with a vision-based feedback system. The focus of expansion, which is equivalent to the robots heading direction, is determined based on acquired features. Variations in the heading direction are detected, integrated and corrected. The accuracy of our drive feedback system has shown to be adequate, but is limited by the preceding feature acquisition.

Additionally, we address the problem of localization in indoor environments. Our method is local, achieving accuracy comparable to commercial indoor localization systems. Translational and rotational movements are determined based on features acquired perpendicular to the mobile robots direction of movement. Our algorithm works with sparse, unstable features and can be run in real-time on a low-cost mobile robot platform.

We finally introduce a novel approach for environment mapping. This approach builds a three-dimensional map using information gathered by the robot sensors. Three-dimensional points are obtained through features, stereo-vision or active-illumination and integrated into a single map representation. We employ our feature acquisition, feature integration and localization method. While stereo vision is computationally too demanding, combining feature based environment mapping with active-illumination stereo-vision generates a usable map for navigating and visualizing real-world indoor environments.

7.2 Directions For Future Work

Although we succeeded to solve some fundamental problems of vision-based mobile robot systems for indoor environment mapping, there are still many issues present that should be addressed in the future.

- *Mobile Robot Platform*

  Technology is advancing quickly and the hardware development cycles get shorter. Hardware components we used in the construction of the mobile robot platform have been updated. The most relevant update occurred to the Apple Mac Mini that we employ as the core unit for computational processing. Besides the improved overall speed, the current model[1] is equipped with 600*Mbit/s* wireless networking functionality. Higher data transfer rates and a broader operating range remove the communication restrictions (Section 2.5), allowing real-time data transfer for supplemental off-platform processing and visualization. The integrated graphics processor supports programmable shaders. Data processing and analysis can thus be drastically improved, freeing resources for more elaborated control algorithms. In particular, stereo-vision could benefit from hardware shaders and perform the disparity map calculation in real-time (Section 6.3). Despite the improved speed, the new features and the faster graphics processor, the power requirements barely changed.

  Our mobile robot platform requires an update to the power electronics. The Li-Ion cells we used in the battery packs are classified as not reliable by the producer and should be replaced (Section 2.1.4). Present Li-Ion cells provide higher capacity at same weight and size factor. Combined with an additional ATX power supply (Section 2.1.5), power peaks can be covered, improving the autonomous operating time and system robustness. To reliably monitor the power and determine the remaining operating time of the mobile robot platform exactly, the current voltage sensor should be replaced with its high precision version.

  One major issue we encountered during our experiments is related to the robots drive system. While the mobile robot platform is rotating, the front-most and the back-most wheels are dragged over the ground perpendicular to their direction of movement. Dependent on the ground surface, a high friction causes abrupt and imprecise movements. This movements result in motion blur, preventing the vision-based environment perception.

  Our sonar-based obstacle detection exposed blind areas, blocking the mobile robot platform in rare cases. A sonar with an expanded cone detection angle would prevent this cases by activating the vision-based refinement.

  Active-illumination in combination with vision-based environment perception showed promising results for the obstacle detection, as well as the map-

[1] Production date late 2009

ping of close-range surfaces. Further improvements could be expected from the use of higher energy lasers (Section 2.4.4), simplifying the laser detection and compensating the continuously decreasing energy output during longer active periods. The laser deflection mechanism uses a low precision servo motor (Section 2.2.3). A stepper motor provides an enhanced resolution, whereby more dense surface representations could be acquired.

- *Environment Adapted Behavior*

  Our obstacle detection system is unable to detect the precise position and shape of obstacles with transparent, reflective or light absorbing surfaces (Section 5.1.4). While transparent and reflective surfaces are the main source of erroneous readings, light absorbing surfaces might not be detected during the vision-based refinement. There is no obvious solution for transparent surfaces, but an increased intensity of the active-illumination could reduce the influence of the obstacle surface properties on the refinement.

  For obstacle avoidance, we consider a dynamic environment and employ a simple strategy that navigates the robot always in the direction of free space. The differentiation between static and dynamic obstacles and the obstacle mapping could help in the development of more advanced obstacle avoidance strategies and prevent navigation dead locks that cannot be solved with our simple approach.

  Furthermore, continuous experiments during our work on feature-based navigation showed that vision-based motion estimation is by far more error-prone when the camera is oriented in parallel to the robot's driving direction. A perpendicular camera orientation (Section 5.3) achieves superior results and hence should be considered for the lateral drift estimation and correction (Section 5.2).

  Illumination has an important impact on the vision-based environment perception. Therefore a system that compensates dynamically for illumination changes would be desirable, adjusting the cameras aperture or acquiring the visual data with multiple exposures.

- *Localization*

  Our mobile robot system uses a local method for self-localization (Section 5.3). The environment position is determined based on a initial position. During navigation this position is continuously updated. Though our localization method is quite accurate, positioning errors accumulate over time. Re-localization could correct these errors. Finding stable landmarks, determining their position with an appropriate precision and correctly identify them inside a real-world indoor environment is still an unsolved problem.

- *Environment Mapping*

  The ambition of environment mapping is to provide a three-dimensional model as complete and accurate as possible from the real-world indoor environment, including color and texture information. This model relies on visual data that is acquired with an autonomously operating system with no prior knowledge and with no human intervention.

  So far this goal can only be achieved with expensive laser-scanners. They are missing the mobility and flexibility of our system which is paramount for unknown environments. We believe that our small low-cost mobile robot system could be an alternative for the three-dimensional environment mapping and further research can resolve the remaining issues.

CHAPTER A

APPENDIX

A.1 Laser classes

Class 1: A class 1 laser is safe to use under all conditions. The maximum permissible exposure (MPE) cannot be exceeded. This class includes high-power lasers within an enclosure that prevents exposure to the radiation and that cannot be opened without shutting down the laser.

Class 2: A class 2 laser is safe because the blink reflex will limit the exposure to no more than 0.25 seconds. It only applies to visible-light lasers (400-700nm). Class-2 lasers are limited to 1mW continuous wave, or more if the emission time is less than 0.25 seconds or if the light is not spatially coherent. Intentional suppression of the blink reflex could lead to eye injury.

Class 1M and 2M: Same restrictions apply as for classes 1 and 2, but the beam of this laser classes should not be viewed through magnifying instruments such as microscopes and telescopes.

Class 3R: Class 3R, is still safe if handled carefully, with restricted beam viewing and a low risk of injury.

Class 3B: Class 3B is hazardous if the eye is exposed directly. Diffuse reflections e.g. from paper or other matte surfaces are not harmful. Protective eyewear is typically required where direct viewing of a class 3B laser beam may

occur. Class 3B lasers must be equipped with a key switch and a safety interlock.

Class 4: Class 4 lasers include all lasers with beam power greater than class 3B. A class 4 laser can burn the skin, in addition to potentially devastating and permanent eye damage as a result of direct or diffuse beam viewing. These lasers may ignite combustible materials, and thus may represent a fire risk.

BIBLIOGRAPHY

[Arun et al., 1987] Arun, K. S., Huang, T. S., and Blostein, S. D. (1987). Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700.

[Ballesta et al., 2007] Ballesta, M., Gil, A., scar Martnez, and scar Reinoso, M. (2007). Local descriptors for visual slam.

[Barshan and Durrant-Whyte, 1994] Barshan, B. and Durrant-Whyte, H. (1994). Orientation estimate for mobile robots using gyroscopic information. In *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*, volume 3, pages 1867–1874 vol.3.

[BatterySpace, 2009] BatterySpace (2009). Batteryspace. Website. Available online at http://www.batteryspace.com/; visited on July 22th 2009.

[Bauer et al., 2007] Bauer, J., Sünderhauf, N., and Protzel, P. (2007). Comparing several implementations of two recently published feature detectors. *Proc. of the International Conference on Intelligent and Autonomous Systems*.

[Bay, 2006] Bay, H. (2006). Surf: Speeded up robust features. Website. Available online at http://www.vision.ee.ethz.ch/ surf/; visited on June 7th 2009.

[Bay et al., 2008] Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded-up robust features. In *9th European Conference on Computer Vision*, Graz, Austria.

[Besl and McKay, 1992] Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256.

[Birchfield, 2007] Birchfield, S. (2007). Klt: An implementation of the kanade-lucas-tomasi feature tracker. Website. Available online at http://www.ces.clemson.edu/ stb/klt/; visited on June 7th 2009.

[Borenstein et al., 1996] Borenstein, J., Everett, H. R., and Feng, L. (1996). where am i? systems and methods for mobile robot positioning. Website. Available online at http://www-personal.umich.edu/ johannb/shared/pos96rep.pdf; visited on January 25th 2010.

[Bradshaw et al., 2007] Bradshaw, J., Lollini, C., and Bishop, B. (2007). On the development of an enhanced optical mouse sensor for odometry and mobile robotics education. *System Theory, 2007. SSST '07. Thirty-Ninth Southeastern Symposium on*, pages 6–10.

[Camus, 1995] Camus, T. A. (1995). Real-time optical flow.

[Chai and Shum, 2000] Chai, J.-X. and Shum, H.-Y. (2000). Parallel projections for stereo reconstruction. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 493–500.

[Chen et al., 2003] Chen, Z., Pears, N., McDermid, J., and Heseltine, T. (2003). Epipole estimation under pure camera translation. In *DICTA*, pages 849–858.

[Conradt, 2008] Conradt, J.-A. (2008). A distributed cognitive map for spatial navigation based on graphically organized place agents.

[Darms et al., 2009] Darms, M. S., Rybski, P. E., Baker, C., and Urmson, C. (2009). Obstacle detection and tracking for the urban challenge. *Trans. Intell. Transport. Sys.*, 10(3):475–485.

[Davison et al., 2007] Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6):1052–1067.

[EvolutionRobotics, 2005] EvolutionRobotics (2005). Northstar. Available online at http://www.evolution.com/; visited on January 25th 2010.

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.

[Forest et al., 2004] Forest, J., Salvi, J., Cabruja, E., and Pous, C. (2004). Laser stripe peak detector for 3d scanners. a fir filter approach. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3*, pages 646–649, Washington, DC, USA. IEEE Computer Society.

[Hagisonic, 2008] Hagisonic (2008). Stargazer. Available online at http://www.hagisonic.com/; visited on January 25th 2010.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151.

[Hartley and Zisserman, 2004] Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

[Hübner and Pajarola, 2009] Hübner, T. and Pajarola, R. (2009). Real-time feature acquisition and integration for vision-based mobile robots. In *ISVC (1)*, pages 189–200.

[Ilstrup and Hugh Elkaim, 2008] Ilstrup, D. and Hugh Elkaim, G. (2008). Low cost, low power structured light based obstacle detection. In *Position, Location and Navigation Symposium, 2008 IEEE/ION*, pages 771–778.

[J. Bauer, 2007] J. Bauer, N. Sünderhauf, P. P. (2007). Comparing several implementations of two recently published feature detectors. *In Proc. of the International Conference on Intelligent and Autonomous Systems*.

[Jonathan and Zhang, 2007] Jonathan, K. and Zhang, H. (2007). Quantitative evaluation of feature extractors for visual slam. In *CRV '07: Proceedings of the Fourth Canadian Conference on Computer and Robot Vision*, pages 157–164, Washington, DC, USA. IEEE Computer Society.

[Karlsson et al., 2005] Karlsson, N., Bernardo, E. D., Ostrowski, J., Goncalves, L., Pirjanian, P., and Munich, M. E. (2005). The vslam algorithm for robust localization and mapping. *Proc. of Int. Conf. on Robotics and Automation (ICRA)*.

[Laboratory, 2009] Laboratory, N. J. P. (2009). Claraty. Website. Available online at http://claraty.jpl.nasa.gov/man/overview/index.php; visited on August 31th 2009.

[Lowe, 1999] Lowe, D. (1999). Object recognition from local scale-invariant features. *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, 2:1150–1157 vol.2.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110.

[MAPIR, 2009] MAPIR (2009). The mobile robot programming toolkit (mrpt). Website. Available online at http://babel.isa.uma.es/mrpt/index.php/Main_Page; visited on September 1th 2009.

[Microsoft, 2009] Microsoft (2009). Robotics developer studio. Website. Available online at http://msdn.microsoft.com/en-us/robotics/default.aspx; visited on August 29th 2009.

[OpenSource, 2008] OpenSource (2008). Fast sift image features library. Website. Available online at http://libsift.sourceforge.net/; visited on June 3th 2009.

[Palacin et al., 2006] Palacin, J., Valgaon, I., and Pernia, R. (2006). The optical mouse for indoor mobile robot odometry measurement. *Sensors and Actuators A: Physical*, 126(1):141 – 147.

[Phidgets, 2009] Phidgets (2009). Phidgets. Website. Available online at http://www.phidgets.com/; visited on July 30th 2009.

[Ramanath et al., 2002] Ramanath, R., Snyder, W. E., Bilbro, G. L., and Iii, W. A. S. (2002). Demosaicking methods for bayer color arrays. *Journal of Electronic Imaging*, 11:306–315.

[Rosten and Drummond, 2006] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443.

[S. Soumare, 2002] S. Soumare, A.Ohya, S. (2002). Real-time obstacle avoidance by an autonomous mobile robot using an active vision sensor and a vertically emitted laser slit. In *Intelligent Autonomous Systems 7, IOS Press (Proceedings of the 7th international confererence on Intelligent Autonomous Systems, California)*, pages 301–308.

[Se et al., 2002] Se, S., Lowe, D., and Little, J. (2002). Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *International Journal of Robotics Research*, 21:735–758.

[Sebastien, 2003] Sebastien, D. (2003). Low overhead optical flow based robot navigation.

[ServoCity, 2009] ServoCity (2009). Servocity. Website. Available online at http://www.servocity.com/; visited on August 11th 2009.

[Shen et al., 2006] Shen, D.-F., Lee, J.-S., Kil, S.-K., Ryu, J.-G., Lee, E.-H., and Hong, S.-H. (2006). 3d reconstruction of scale-invariant features for mobile robot localization. *IJCSNS International Journal of Computer Science and Network Security*, 6(3B):101–109.

[Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600.

[Smith and Brady, 1997] Smith, S. M. and Brady, J. M. (1997). Susan—a new approach to low level image processing. *Int. J. Comput. Vision*, 23(1):45–78.

[Source, 2009] Source, O. (2009). Libdc1394. Website. Available online at http://sourceforge.net/projects/libdc1394/; visited on September 8th 2009.

[Source, 2010] Source, O. (2010). avcap. Website. Available online at http://sourceforge.net/projects/libavcap/; visited on March 11th 2010.

[Srinivasan et al., 1996] Srinivasan, M., Zhang, S., Lehrer, M., and Collett, T. (1996). Honeybee navigation en route to the goal: visual flight control and odometry. *The Journal of Experimental Biology*, 199(1):237–244.

[Tellzer, 2001] Tellzer, S. (2001). Optical flow based robot navigation.

[Teschner et al., 2003] Teschner, M., Heidelberger, B., Mueller, M., Pomeranets, D., and Gross, M. (2003). Optimized spatial hashing for collision detection of deformable objects. *Proceedings of Vision, Modeling, Visualization (VMV 2003)*, pages 47–54.

[Tomasi and Kanade, 1991] Tomasi, C. and Kanade, T. (1991). Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University.

[Triggs et al., 2000] Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (2000). Bundle adjustment - a modern synthesis. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pages 298–372, London, UK. Springer-Verlag.

[Ulrich and Nourbakhsh, 2000] Ulrich, I. and Nourbakhsh, I. R. (2000). Appearance-based obstacle detection with monocular color vision. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 866–871. AAAI Press / The MIT Press.

[Umeyama, 1991] Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376–380.

[Viet and Marshall, 2007] Viet, C. N. and Marshall, I. (2007). Vision-based obstacle avoidance for a small, low-cost robot. In *ICINCO 2007, Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics, Robotics and Automation 1, Angers, France, May 9-12, 2007*, pages 275–279.

[Wei et al., 2009] Wei, B., Gao, J., Li, K., Fan, Y., Gao, X., and Gao, B. (2009). Indoor mobile robot obstacle detection based on linear structured light vision system. In *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pages 834–839.

[Zhang and Kanade, 1998] Zhang, Z. and Kanade, T. (1998). Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27:161–195.

CURRICULUM VITAE

Personal Information

Name	Thomas Hübner
Date of birth	November 20, 1974
Place of birth	Weißenfels, Germany

Education

2005 - 2010 Doctor in Informatics in the Visualization and Multimedia Lab, Department of Informatics, University of Zurich, Switzerland
Subject of dissertation: A REAL-TIME VISION-BASED MOBILE ROBOT SYSTEM
Advisor: Prof. Dr. R. Pajarola
Co-Examiner: Prof. A. Bernstein, Ph.D.

2001 - 2004 Medical Application Developer, Department of Neuroradiology, University Hospital Bern, Switzerland

2001	Master of Science, Department of Computer Science, Otto-von-Guericke University Magdeburg, Germany Subject of master thesis: ”Multiplanar Reformation of MR- and CT-images for treatment planning in implant dentistry” Advisor: Prof. Dr. Klaus D. Tönnies
1999 - 2001	Student of Computational Visualistics at the Otto-von-Guericke University Magdeburg, Germany
1999	Dipl. Informatik, Department of Applied Computer Science, University Fulda, Germany Subject of diploma thesis: ”Three-dimensional visualization with stereoscopic systems in OpenGL” Advisor: Prof. Dr. Werner Heinzel
1993 - 1999	Student of Computer Science at the University Fulda, Germany
1991 - 1993	Goethe Gymnasium Weißenfels, Germany

Publications

Conference Publications

T. Hübner and R. Pajarola, Real-time Feature Acquisition and Integration for Vision-based Mobile Robots and Scanning Systems, Proceedings International Symposium on Visual Computing, pages 189-200, 2009.

T. Hübner and R. Pajarola, Real-time Vision-based Lateral Drift Correction, Proceedings EUROGRAPHICS Short Papers, pages 13-16, 2009.

T. Hübner and R. Pajarola, Single-Pass Multi-View Volume Rendering, Proceedings IADIS International Conference Computer Graphics and Visualization, pages 50-58, 2007.

T. Hübner, Y. Zhang, R. Pajarola, Multi-View Point Splatting, Proceedings Conference on Computer Graphics and Interactive Techniques in Australasia and South-East Asia (GRAPHITE), pages 285-294, 2006.

T. Hübner, H. Oswald, G. Schroth, Multiplanar Reformation of MR- and CT-images for treatment planning in implant dentistry, Computer Assisted Radiology

and Surgery, 2004 .

M. Hinz and R. Pohle and T. Hübner and K. D. Tönnies, 3D-Analyse medizinischer Volumendaten unter Nutzung automatisch generierter Transferfunktionen., Bildverarbeitung für die Medizin, pages 290-294, 2001.

Journal Articles

T. Hübner, Y. Zhang, R. Pajarola, Single-Pass Multi-View Rendering, IADIS International Journal on Computer Science and Information System, pages 122-140, 2007.

Misc

L. Remonda, K. Dula , T. Hübner, C. Kiefer , G. Schroth, D. Buser, Dental MR imaging: comparison with Dental CT imaging for treatment planning in implant dentistry., 41th Annual meeting of the ASNR, Washington, USA, 2003.

L. Remonda, K. Dula , T. Hübner, C. Kiefer , G. Schroth, D. Buser, Dental MR imaging: comparison with Dental CT imaging for treatment planning in implant dentistry., VVIIth Symposium Neuroradiologicum, Paris, France, 2003.

Awards

Best paper award, Single-Pass Multi-View Volume Rendering, IADIS International Conference Computer Graphics and Visualization, 2007